仅发 Linux 伊甸园

鸟哥的 Linux 私房菜

文档收集：檸檬ｄë單純

尊重作者鸟哥,转载请示出处!

交流群：7744707 *Welcome You!!!*

什么是 Linux

最近更新日期：2006/05/31

众所皆知的， Linux 的核心原型是 1991 年由托瓦兹 (Linus Torvalds) 写出来的， 但是托瓦兹为何可以写出 Linux 这个操作系统？为什么他要选择 386 的计算机来开发？ 为什么 Linux 的发展可以这么迅速？又为什么 Linux 是免费的？ 以及目前为何有这么多的 Linux 版本 (distributions) 呢？ 了解这些东西后，才能够知道为何 Linux 可以免除专利软件之争， 并且了解到 Linux 为何可以同时在个人计算机与大型主机上面大放异彩！ 所以，在实际进入 Linux 的世界前，就让我们来谈一谈这些有趣的历史故事吧！ ^_^

什么是 Linux ？

Linux 这玩意儿是在计算机上面运作的，所以，当然得要了解一下计算机这玩意儿。首先，到底有哪些种类的计算机呢？ 而 Linux 可以在哪些种类的计算机上面运作？且 Linux 源自哪里？这些我们都得来谈一谈先！

计算器：计算的辅助工具

在目前的都市生活中，您应该很难不接触到计算机这个玩意儿吧？ 这个计算机当初在开发的时候，是希望可以辅助与简化人们进行大量的运算工作， 后来才发展成为一些特殊用途。无论如何，计算机基本的功能就是：『接受使用者输入指令，经由 CPU 的数学与逻辑单元运算处理后， 以产生或储存成有用的信息』。为了达成这个功能，当然计算机就必须要有：

- 输入单元：例如鼠标、键盘、卡片阅读机等等

- 中央处理器(CPU)：含有算数逻辑、控制、记忆等单元
- 输出单元：例如屏幕、打印机等等

如果您看过计算机的话，那么上面的东西其实就是组成您计算机的主要组件啰！ 而为了连结各个组件，因此有了主机板，所以，您的主机里面就包含了主机板以及 CPU ， 还有各种需要的适配卡。而屏幕、键盘、鼠标则透过与主机的连结，就构成一部可以运作的计算机了。 另外，由于计算机仅认识 0/1 ，因此计算机主要是以二进制的方式来计算的， 因此，通常计算机的记忆/储存单位都是以 Byte 或 bits 为基本单位。他们的单位是这样的：

- 1 Bytes = 8 bits
- 1 KB = 1024 Bytes
- 1 MB = 1024 KB
- 1 GB = 1024 MB

而计算机也因为他的复杂度，而分为数种等级，例如：

- 超级计算机(Supercomputer)：
  超级计算机是运作速度最快的计算机，但是他的维护、操作费用也最高！ 主要是用于需要有高速计算的计划中。例如：国防军事、气象预测、太空科技， 用在模拟的领域较多。详情也可以参考：国家高速网络与计算中心http://www.nchc.org.tw/ 的介绍！ 至于全世界的 500 大超级计算机，则请参考：http://www.top500.org/
- 大型计算机(Mainframe Computer)：
  大型计算机通常也具有数个高速的 CPU ，功能上虽不及超级计算机， 但也可用来处理大量资料与复杂的运算。例如大型企业的主机、 全国性的证券交易所等每天需要处理数百万笔数据的企业机构， 或者是大型企业的数据库服务器等等。
- 迷你计算机(Minicomputer)：
  迷你计算机仍保有大型计算机同时支持多使用者的特性，但是主机可以放在一般作业场所， 不必像前两个大型计算机需要特殊的空调场所。通常用来作为科学研究、 工程分析与工厂的流程管理等。
- 微电脑(Microcomputer)：
  又可以称为个人计算机，也是我们这本书主要探讨的目标！体积最小，价格最低， 但功能还是五脏俱全的！大致又可分为桌上型、笔记型等等。

虽然在目前个人计算机的使用甚为广泛，但是在 1990 年以前，个人计算机是比较不被重视的！ 因为(1)他的运算速度在当时实在很差，而且(2)当时比较有名的操作系统也没有对个人计算机支持。 所以才会流行不太起来～ 嘿嘿！提到操作系统啰～ 底下我们就来谈一谈之前的操作系统。

什么是操作系统

什么是操作系统 (Operation System, OS) 呢？我们先来想一想， 上面介绍的计算器(计算机)是如何工作的？举例来说，您计算机屏幕上面显示的讯息， 是如何显示出来的呢？嗯！是藉由显示卡与屏幕显像的。那么如果你想要看 VCD 呢？ 呵呵，就需要 1.)有影音数据的光盘片、 2.)可读取光盘片的光驱、 3.) 可以转换影音数据输出的中央处理器 (CPU)、 4.)可以显示影像的显示芯片(显示卡)、 5.) 可以传输声音的音效芯片(声卡)、 6.)可以输出影像的屏幕以及 7.)可以发出声音的喇叭！ 也就是说：所有在『工作』的东西都是『硬件』呀！ 对啦！就是硬件在工作！

那么问题来了，现在我们知道，计算机所进行的工作都是计算机硬件帮我们达成的，但是， 为什么这些硬件知道如何播放 VCD 呢？这当然是因为有某个东西在正确的控制硬件的工作了， 那个咚咚就是： 操作系统 啦！ 操作系统可以管理整部计算机的硬件，他可以控制 CPU 进行正确的运算， 他可以分辨硬盘里头的数据并进行读取，他还必须要能够认识所有的适配卡， 这样，才能够将所有的硬件通通正确的使用上啊！所以，如果没有这个操作系统，那么您的计算机就等于是一堆废铁而已啊！

虽然操作系统可以完整的掌控所有的硬件资源，但是，对于使用者来说，还是不够的！ 因为操作系统虽然可以掌控所有的硬件，但是，如果使用者无法与操作系统沟通， 那么这个操作系统就没有什么用处了。简单的来说，以上面的 VCD 为例， 虽然操作系统可以控制硬件播放 VCD ，但是，如果使用者没有办法控制何时要播出 VCD 的话， 那么到底我们要怎么看 VCD 啊？对吧！

所以说，一个比较『完整的操作系统』应该要包含 两个东西，一个是『核心与其提供的接口工具』、 一个是『利用核心提供的接口工具所开发出来的软件』。 我们以大家常使用的 Windows 计算机来做一个简单的说明好了。大家应该都使用过 Windows 计算机里面的『档案总管』吧！当你开启档案总管的时候，他就会显示你硬盘当中的数据， 这个『显示硬盘里面的数据，就是核心帮你做的』，但是， 『你要核心去显示硬盘哪一个目录下的数据，则是由档案总管这个工具帮你达成的』！ 这样可以理解吗？

那么核心有没有作不到的事？当然有的，举例来说， 如果您曾经自行安装过比较新的显示卡在您的个人计算机上面， 那么应该常常会发生 Windows 计算机告知您:『找不到合适的驱动程序来显示』的问题吧？ 也就是说，即使您有最新的显示卡安装在您的个人计算机上面，而且也有播放 VCD 的程序， 但是因为『核心』无法操控这个最新的显示卡，所以，就无法正常的显示您的 VCD 了。 没错！你的整个硬件是由核心来管理的，而如果核心不认识你的硬件， 那么你将无法使用该硬设备，例如上面提到的最新的显示卡。

> Tips:
> 在定义上，只要能够让计算机硬件正确无误的运作，那就算是操作系统了。所以说，操作系统其实就是核心与其提供的接口工具，不过，就如同上面讲的，因为最阳春的核心缺乏了与使用者沟通的亲和接口， 所以在目前，一般我们提到的『操作系统』都会包含核心与相关的使用者应用软件呢！



核心就是『Kernel』， 他是一个操作系统的最底层的东西，由他来掌管整个硬件资源的工作状态，而 每个操作系统都有自己的核心，所以说， 当有新的硬件加入到你的系统中的时候，若你的『 Kernel 』并没有支援他的时候，呵呵，这个新的硬件就肯定无法工作的，因为控制他的 Kernel 并不认识他呀！这样了解了吗？！先有个概略性的了解， 后面我们提到『核心编译』的时候会再更详细的谈到他！

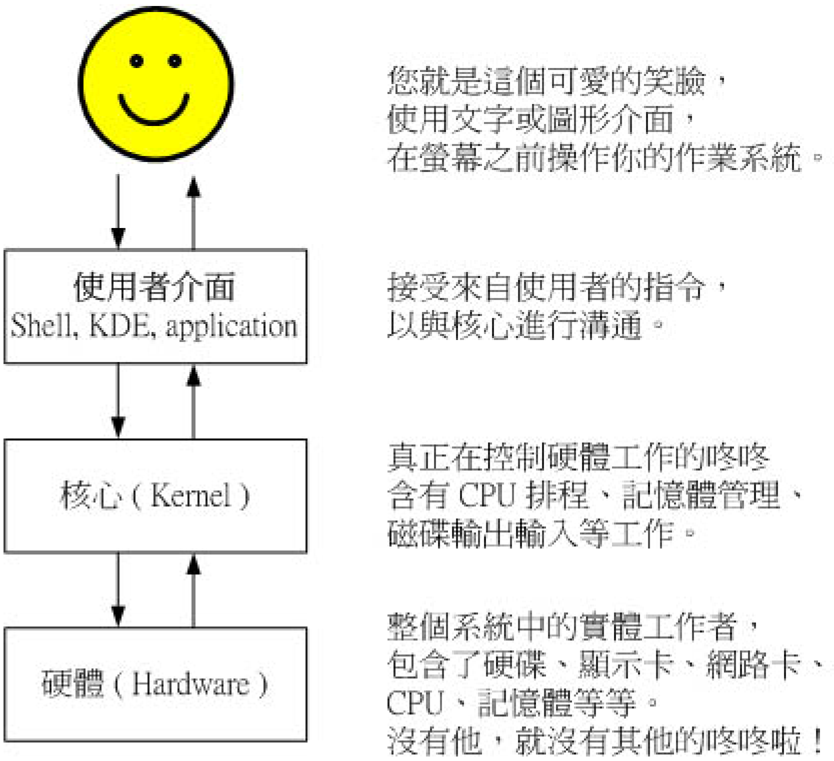
一般来说，Kernel 为了达成使用者所需要的正确运算结果，他必须要管理的事项有：

- 系统呼叫接口(System call interface) ： 为了方便程序开发者可以轻易的透过与 kernel 的沟通，将硬件的资源进一步的利用， 于是需要有这个简易的接口来方便程序开发者。
- 行程管理(Process control) ： 总有听过所谓的『多任务环境』吧？一部计算机可能同时间有很多的工作跑到 CPU 等待运算处理， Kernel 这个时候必须要能够控制这些工作，让 CPU 的资源作有效的分配才行！
- 内存管理(Memory management) ： 控制整个系统的内存管理，若内存不足， Kernel 最好还能够提供虚拟内存的功能！

- 档案系统管理(File system management) ： 档案系统的管理，例如数据的输入输出 (I/O) 等等的工作啦！ 还有不同档案格式的支持啦等等，如果你的核心不认识某个档案系统， 那么您将无法使用该档案格式的档案啰！例如： Windows 98 就不认识 NTFS 档案格式的硬盘；
- 装置的驱动(Device drivers) ： 就如同上面提到的，硬件的管理是 Kernel 的主要工作之一，当然啰，装置的驱动程序就是核心需要做的事情啦！ 好在目前都有所谓的『可加载模块』功能，可以将驱动程序编辑成模块， 就不需要重新的编译核心啦！这个也会在后续的核心编译当中提到的！

所以啦！所有硬件的资源都是 kernel 来管理的！至于我们要达成一些工作时，除了藉由核心本身提供的功能 (例如上面提到的档案总管) 之外，还可以藉由其它的应用软件来达成喔！举个例子来说，你要看 VCD 影片是吧!那么除了 Windows 提供的媒体播放程序之外,你也可以自行安装 VCD 播放程序来播放 VCD 啰！这个播放程序就是应用软件啰，而这个应用软件可以帮你去控制核心来工作 (就是放映影片啰) ，因此，我们可以这样说， 核心是控制整个硬件支持的咚咚，也是一个操作系统的最底层， 然而要让整个操作系统更完备的话，那还需要含有相当丰富的核心提供的工具， 以及核心相关的应用软件来支持。

OK！提到这里那么您知道 Linux 是什么了吗？呵呵！对啦！其实 Linux 就是一个操作系统，这个操作系统里头含有最主要的 kernel 以及 kernel 提供的工具啦！他提供了一个完整的操作系统当中最底层的硬件控制与资源管理的完整架构， 这个架构是沿袭 Unix 良好的传统来的，所以相当的稳定而功能强大！此外， 由于这个优良的架构可以在目前的个人计算机(X86 系统) 上面跑， 所以很多的软件开发者将他们的工作心血移转到这个架构上面，那就是很多的应用软件啦！ 虽然 Linux 仅是其核心与核心提供的工具，不过，由于核心、核心工具与这些软件开发者提供的软件的整合，使得 Linux 成为一个更完整的、功能强大的操作系统啰！我们可以将 Linux 的系统与使用者的相关性看成底下的图示：



图一、使用者、使用者接口与核心工具、核心、与硬件之相关性

约略了解 Linux 是何物之后，接下来，我们要谈一谈，『为什么说 Linux 是很稳定的操作系统呢？他是如何来的？』

Tips:
Torvalds 先生在写出 Linux 的时候，其实该核心仅能『驱动 386 所有的硬件』而已， 所谓的『让 386 计算机开始运作，并且等待使用者指令输入』而已，事实上， 当时能够在 Linux 上面跑得软件还很少呢！



Tips:
由上面的说明中，我们知道硬件是由『核心』来控制的，而每种操作系统都有他自己的核心。 这就产生了一个很大的问题，因为早期硬件的开发者所开发的硬件架构或多或少都不相同， 举例来说，2006 年以前的麦金塔是请 IBM 公司开发自己的硬件与操作系统，Windows 则是开发在 x86 架构上的操作系统之一， 那么 Windows 是否可以在麦金塔上面跑？答案是『不行』的！不过，在 2006 年以后， 麦金塔转而请 Intel 设计其硬件架构，亦即其硬件架构已经转为 x86 系统，因此在 2006 年以后的麦金塔若使用 x86 架构时，其硬件则『可能』可以安装 Windows 操作系统了。



Tips:
因为 Windows 操作系统本来就是针对个人计算机 x86 架构的硬件去设计的，所以他当然只能在 x86 的个人计算机上面运作，在不同的平台，当然就无法运行了。也就是说， 每种操作系统都是在他专门的机器上面运行的喔!这点得要先了解。不过，Linux 由于是 Open Source 的操作系统，所以他的程序代码可以被修改成适合在各种机器上面运行的， 也就是说， Linux 是具有『可移植性』，这可是很重要的一个功能喔！ ^_^



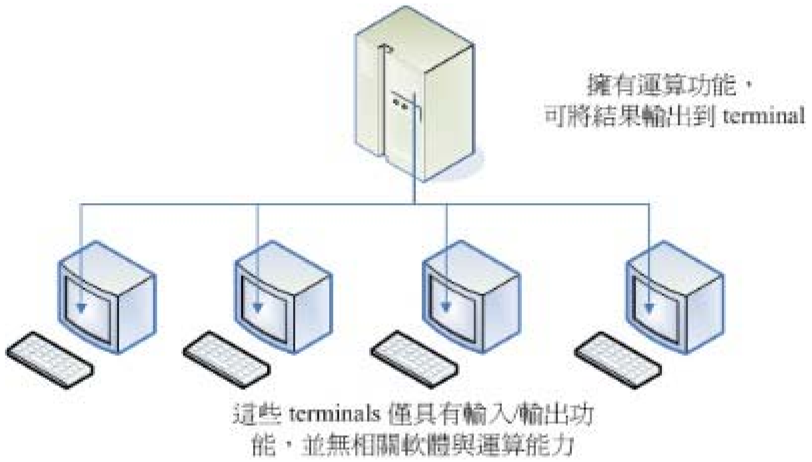
Linux 之前， Unix 的历史

早在 Linux 出现之前的二十年 (大约在 1970 年代), 就有一个相当稳定而成熟的操作系统存在了！那就是 Linux 的老大哥『 Unix 』是也！ 怎么这么说呢？！他们这两个家伙有什么关系呀？这里就给他说一说啰！众所皆知的， Linux 的核心是由 Linus Torvalds 在 1991 年的时候给他开发出来的， 并且丢到网络上提供大家下载，后来大家觉得这个小东西 (Linux Kernel) 相当的小而精巧，所以慢慢的就有相当多的朋友投入这个小东西的研究领域里面去了！ 但是为什么这的小东西这么棒呢？！然而又为什么大家都可以免费的下载这个东西呢？！ 嗯！等鸟哥慢慢的唬 xx....喔不！听我慢慢的道来！

- 1969 年以前：一个没有完成的梦想： Bell, MIT 与 GE 的『 Multics』系统

  早期的计算机并不像现在的个人计算机一样，他可不是一般人碰的起的呢～ 除非是军事或者是高科技用途，或者是学术单位的学术研究，否则，真的很难接触到。 非但如此，早期的计算机架构还很难使用，除了运算速度并不快之外，操作接口也很困扰的！ 在那个时候，写程序是件很可怜的事情，因为，程序设计者，必须要将程序相关的信息在读卡纸上面打洞， 然后再将读卡纸插入卡片阅读机来将信息读入主机中运算。光是这样就很麻烦了，如果程序有个小地方写错， 哈哈！光是重新打卡就很惨，加上主机少，使用者众多，光是等待，就耗去很多的时间了！

在之后，经由操作系统的改良，使得后来可以使用键盘来进行信息的输入/输出。 不过，在一间学校里面，主机毕竟可能只有一部，如果多人等待使用，那怎么办？ 好在 1960 年代初期麻省理工学院 (MIT) 发展了所谓的： 『兼容分时系统(Compatible Time-Sharing System, CTSS)』，它可以让大型主机透过提供数个终端机(terminal)以联机进入主机，来利用主机的资源进行运算工作。 架构有点像这样：



图二、早期主机与终端机的相关性图标

如此一来，无论主机在哪里，只要在终端机前面进行输入输出的作业，就可利用主机提供的功能了。 不过，需要注意的是，此时终端机只具有输入/输出的功能，本身完全不具任何运算或者软件安装的能力。 而且，比较先进的主机大概也只能提供 30 个不到的终端机而已。

为了更加强化大型主机的系统，以让主机的资源可以提供更多使用者来利用，所以在 1965 年前后， 由贝尔实验室 (Bell)、麻省理工学院 (MIT) 及奇异公司 (GE) 共同发起了 Multics 的计划，Multics 目的是想要让大型主机可以达成提供 300 个以上的终端机联机使用的目标。不过，到了 1969 年前后，计划进度落后，资金也短缺，所以该计划就宣告不治.....喔！是宣告失败～ (注： Multics 有复杂、多数的意思存在。)

- 1969 年： Ken Thompson 的小型 file server system

  在认为 Multics 计划不可能成功之后，贝尔研究室 就退出该计划。不过，原本参与 Multics 计划的人员中，已经从该计划当中获得一些点子， Ken Thompson 就是其中一位！Thompson 因为自己的需要，希望开发一个小小的操作系统， 以提供自己的需求。在开发时，有一部 DEC (Digital Equipment Corporation) 的 PDP-7 没人使用， 于是他就准备针对这部主机进行操作系统核心程序的撰写。本来 Thompson 是没时间的， 有趣的是，在 1969 年八月份左右，刚好 Thompson 的妻儿去了美西探亲， 于是他有了额外的一个月的时间好好的待在家将一些构想实现出来！ 经过四个星期的奋斗，他终于以组译语言 (Assembler) 写出了一个核心程序，同时包括一些核心工具程序， 以及一个小小的档案系统。那个系统就是 Unix 的原型！ 当时 Thompson 将 Multics 庞大的复杂系统简化了不少，于是同实验室的朋友都戏称这个系统为： Unics 。 Thompson 的这个档案系统有两个重要的概念，分别是：

  - 所有的程序或系统装置都是档案
  - 不管建构编辑器还是附属档案，所写的程序只有一个目的，且要有效的完成目标。

这些概念在后来对于 Linux 的发展有相当重要的影响喔！

- 1973 年： Unix 的正式诞生，Ritchie 等人以 C 语言写出第一个正式 Unix 核心

  由于 Thompson 写的那个操作系统实在太好用了，所以在贝尔实验室内部广为流传，并且数度经过改版。 但是，比较重要的改版则发生在 1973 年。Unix 本来是以组译语言写成的，后来因为系统移植与效能的需求， 该系统被 B 语言所改写。不过，效能依旧不是很好。后来， Dennis Ritchie 将 B 语言重新改写成 C 语言，C 语言算是比较高阶的程序语言，可以在不同的机器上面运作， 而 Ritchie 等人也同时将原本 Thompson 的那个操作系统重新以 C 语言改写， 最后发行出 Unix 的正式版本！

  在这个时候需要特别注意的是，贝尔实验室是隶属于 AT&T 的，只是 AT&T 当时忙于其它商业活动， 所以对于 Unix 是采取比较开放的态度，此外， Unix 在这个时期的发展者都是贝尔实验室的工程师， 这些工程师对于程序当然相当有研究，所以， Unix 在此时当然是不容易被一般人所接受的！ 此外，也需要特别强调，由于 Unix 是以较高阶的 C 语言写的，相对于组译语言需要与硬件有密切的配合， 高阶的 C 语言与硬件的相关性就没有这么大了！所以，这个改变也使得 Unix 很容易被移植到不同的机器上面喔！
- 1977 年： 重要的 Unix 分支： BSD 的诞生

  前面说到，虽然贝尔属于 AT&T ，但是 AT&T 此时对于 Unix 是采取开放的态度， 此外，Unix 是以高阶的 C 语言写成的，理论上是具有可移植性的！ 所以，只要取得 Unix 的原始码，并且针对大型主机的特性加以修订原有的原始码(Source Code)， 就可能将 Unix 移植到另一部不同的主机上头了。所以在 1973 年以后， Unix 便得以与学术界合作开发！最重要的接触就是与加州柏克莱 (Berkeley) 大学的合作了。 柏克莱大学的 Bill Joy 在取得了 Unix 的核心原始码后，着手修改成适合自己机器的版本， 并且同时增加了很多工具软件与编译程序，最终将他命名为 Berkeley Software Distribution (BSD) 。这个 BSD 是 Unix 很重要的一个分支， Bill Joy 也是 Unix 业者『 Sun 』这家公司的创办者！ Sun 公司即是以 BSD 发展的核心进行自己的商业 Unix 版本的发展的。 (后来可以安装在 x86 硬件架构上面 FreeBSD 即是 BSD 改版而来！)
- 1979 年： 一个措手不及的版权宣告！

  由于 Unix 的高度可移植性与强大的效能，加上当时并没有版权的纠纷， 所以让很多商业公司开始了 Unix 操作系统的发展，例如 AT&T 自家的 System V、IBM 的 AIX 以及 HP 与 DEC 等公司，都有推出自家的主机搭配自己的 Unix 操作系统。

  但是，如同我们前面提到的，操作系统的核心 (Kernel) 必须要跟硬件配合， 以提供及控制硬件的资源进行良好的工作！ 而在早期每一家生产计算机硬件的公司还没有所谓的『协议』的概念，所以每一个计算机公司出产的硬件自然就不相同啰！因此他们必须要为自己的计算机硬件开发合适的 Unix 系统。 例如在学术机构相当有名的 Sun 、 Cray 与 HP 就是这一种情况。 他们开发出来的 Unix 操作系统以及内含的相关软件并没有办法在其它的硬件架构下工作的！且由于没有厂商针对个人计算机设计 Unix 系统，因此，在早期并没有支持个人计算机的 Unix 操作系统的出现 (由于 Unix 强调的是多人多任务的环境，但早期的 x86 个人计算机架构下的 CPU 是没有能力达到多任务的作业，因此，并没有人对移植 Unix 到 x86 的计算机上有兴趣)。 每一家公司自己出的 Unix 虽然在架构上面大同小异，但是却真的仅能支持自身的硬件， 所以啰，早先的 Unix 只能与服务器 (Server) 或者是大型工作站 (Workstation) 划上等号！

  但是这个高度开放的 Unix 系统在 1979 年有了重大的转折～ 因为 AT&T 由于商业的考虑，以及在当时现实环境下的思考，于是将想 Unix 的版权收回去， 因此，在 AT&T 在 1979 年发行的第七版 Unix 中，特别提到了 『不可对学生提供原始码』的严格限制！ 同时，也造成 Unix 业界之间的紧张气氛，并且也引爆了很多的商业纠纷～

- 1984 年之一： x86 架构的 Minix 诞生

  关于 1979 年的版权声明中，影响最大的当然就是学校教 Unix 相关学问的教授了！ 想一想，如果没有核心原始码，那么如何教导学生认识 Unix 呢？ 这问题对于 Andrew Tanenbaum (谭宁邦) 教授来说，实在是很伤脑筋的！ 不过，学校的课程还是得继续啊！那怎么办？？ 既然 1979 年的 Unix 第七版可以在 Intel 的 x86 架构上面进行移植， 那么是否意味着可以将 Unix 改写并移植到 x86 上面了呢？在这个想法上， 谭宁邦教授于是乎自己动手写了 Minix 这个 Unix Like 的核心程序！ 在撰写的过程中，为了避免版权纠纷，谭宁邦完全不看 Unix 核心原始码！ 并且强调他的 Minix 必须能够与 Unix 兼容才行！谭宁邦在 1984 年开始撰写核心程序， 到了 1986 年终于完成，并于次年出版 Minix 相关书籍，同时与新闻群组相结合～

  这个 Minix 版本比较有趣的地方是，他并不是完全免费的，无法在网络上提供下载！ 必须要透过磁盘/磁带购买才行！虽然真的很便宜～不过，毕竟因为没有在网络上流传， 所以 Minix 的传递速度并没有很快速！此外，购买时，随磁盘还会附上 Minix 的原始码！ 这意味着使用者可以学习 Minix 的核心程序设计概念喔！ (这个特色对于 Linux 的启始开发阶段，可是有很大的关系喔！) 此外，开发者仅有谭宁邦教授，因为学者很忙啊！加上谭宁邦始终认为 Minix 主要用在教育用途上面， 所以对于 Minix 是点到为止！所以， Minix 很受欢迎没错， 不过，使用者的要求/需求的声音可能就比较没有办法上升到比较高的地方了！ 这样说，您明白吧？？？ ^_^
- 1984 年之二： GNU 与 FSF 计划的成立

  Richard Mathew Stallman (史托曼) 在 1984 年发起的 GNU 计划，对于现今的自由软件风潮， 真有不可磨灭的地位！目前我们所使用得很多自由软件，几乎均直接或间接帮助于 GNU 这个计划呢！那么史托曼是何许人也？为何他会发起这个 GNU 计划呢？

  Richard Mathew Stallman (生于 1953 年， 网络上自称的 ID 为 RMS) 从小就很聪明！他在 1971 年的时候，进入黑客圈中相当出名的人工智能实验室(AI Lab.)， 这个时候的黑客专指计算机功力很强的人，而非破坏计算机的怪客 (cracker) 喔！ 当时的黑客圈对于软件的着眼点几乎都是在『分享』，所以并没有专利方面的困扰！ 这个特色对于史托曼的影响很大！不过，后来由于管理阶层的问题，导致实验室的优秀黑客离开该实验室， 并且进入其它商业公司继续发展优秀的软件。但史托曼并不服输，仍然持续在原来的实验室开发新的程序与软件。 后来，他发现到，自己一个人并无法完成所有的工作，于是想要成立一个开放的团体来共同努力！

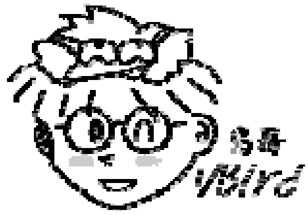
  1983 年以后，因为实验室硬件的更换，使得史托曼无法继续以原有的硬件与操作系统继续自由程序的撰写～ 而且他进一步发现到，过去他所使用的 Lisp 操作系统，是麻省理工学院的专利软件，是无法共享的，这对于想要成立一个开放团体的史托曼是个阻碍。于是他便放弃了 Lisp 这个系统。 后来，他接触到 Unix 这个系统，并且发现， Unix 在理论与实际上，都可以在不同的机器间进行移植。 于是他开始转而使用 Unix 系统。因为 Lisp 与 Unix 是不同的系统， 所以，他原本已经撰写完毕的软件是无法在 Unix 上面运行的！为此，他就开始将软件移植到 Unix 上面。并且，为了让软件可以在不同的平台上运作， 因此，史托曼将他发展的软件均撰写成可以移植的型态！

  1984 年，史托曼开始 GNU 计划，这个计划的目的是想要： 建立一个自由的开放的 Unix 操作系统。 但是建立一个操作系统谈何容易啊！而且在当时的 GNU 是仅有自己一个人单打独斗的史托曼～ 这实在太麻烦，但又不能不做这个计划，于是史托曼反其道而行～『既然操作系统太复杂，我就先写可以在 Unix 上面运行的小程序，这总可以了吧？呵呵！』在这个想法上， 史托曼便开始了程序的写作。在写作期间，为了不让自己吃上官司，他绝对不看专利软件的原始码！ 为了这

个计划，他开始使用原本 Unix 上面跑的软件， 并自行撰写功能与 Unix 原有专利软件相仿的软件。

但不论是什么软件，都得要进行编译成为二进制档案(binary file)后才能够执行， 因此他便开始撰写 C 语言的编译器，那就是现在相当有名的 GNU C (gcc)！ 这个点相当的重要！这是因为 C 语言编译器版本众多，但都是专利软件， 如果他写的 C 编译器够棒，效能够佳，那么将会大大的让 GNU 计划出现在众人眼前！

Tips:
我们在前面稍微提过，计算机仅认识 0/1 的数据，但是人类不认识啊！ 人类对于纯文字的数据(就是所谓的 ASCII 档案格式)比较有感觉。 但是偏偏计算机又不认识 ASCII 格式的文字，很头痛，不是吗？ 为此，就会有很多的所谓的『编译器』来辅助我们撰写程序。 我们一般使用文字编辑器以 ASCII 纯文字格式来撰写程序， 再透过所谓的『编译器(compiler)』』将刚刚完成的文本文件 『编译』成为计算机认识的二进制制(binary file)的档案， 以让计算机认识且可以执行的程序啊！



但开始时并不顺利，为此，他先转而将 Emacs 编辑器写成可以在 Unix 上面跑得软件， 并公开公布原始码，因为 Emacs 太优秀了，因此，很多人便直接向他购买。此时 Internet 尚未流行，所以，史托曼便借着 Emacs 以磁带(tape)出售，赚了一点钱 ，进而开始全力撰写其它软件。并且成立 自由软件基金会 (FSF, Free Software Foundation)，请更多工程师与志工撰写软件。 终于还是完成了 GCC，这比 Emacs 还更有帮助！此外，他还撰写了更多可以被呼叫的 C 函式库(GNU C library)， 以及可以被使用来操作操作系统的基本接口 BASH shell ！ 这些都在 1990 年左右完成了！

Tips:
如果纯粹使用文字编辑器来编辑程序的话，那么程序语法如果写错时， 只能利用编译时发生的错误讯息来修订了，这样实在很没有效率。 Emacs 则是一个很棒的编辑器！注意！是编辑(editor)而非编译(compiler)！ 他可以很快的立刻显示出您写入的语法可能有错误的地方，这对于程序设计师来说， 实在是一个好到不能再好的工具了！所以才会这么的受到欢迎啊！



到了 1985 年，为了避免 GNU 所开发的自由软件被其它人所利用而成为专利软件， 所以他与律师草拟了有名的通用公共许可证 (General Public License, GPL)， 并且称呼他为 copyleft (相对于专利软件的 copyright！)。 关于 GPL 的相关内容我们在下一个小节继续谈论，在这里，必须要说明的是， 由于有 GNU 所开发的几个重要软件，如：

- Emacs
- GNU C (GCC)
- GNU C Library (glibc)
- Bash shell

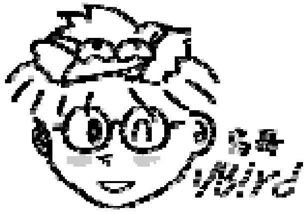
造成后来很多的软件开发者可以藉由这些基础的工具来进行程序开发！ 进一步壮大了自由软件团体！这是很重要的！不过，对于 GNU 的最初构想 『建立一个自由的 Unix 操作系统』来说，有

这些优秀的程序是仍无法满足， 因为，当下并没有『自由的 Unix 核心』存在.....所以这些软件仍只能在那些有专利的 Unix 平台上工作～～一直到 Linux 的出现......

- 1988 年： 图形接口 XFree86 计划

  有鉴于图形使用者接口 (Graphical User Interface, GUI) 的需求日益加重，在 1984 年由 MIT 与其它协力厂商首次发表了 X Window System ，并且更在 1988 年成立了非营利性质的 XFree86 这个组织。所谓的 XFree86 其实是 X Window System + Free + x86 的整合名称呢！ 而这个 XFree86 的 GUI 界面更在 Linux 的核心 1.0 版于 1994 年释出时，整合于 Linux 操作系统当中！

  Tips:
  为什么称图形使用者接口为 X 呢？因为由英文单字来看，Window 的 W 接的就是 X 啦！意指 Window 的下一版就是了！ 需注意的是， X Window 并不是 X Windows 喔！



- 1991 年： 芬兰大学生 Linus Torvalds 的一则简讯

  到了 1991 年，芬兰的赫尔辛基大学的 Linus Torvalds 在 BBS 上面贴了一则消息， 宣称他以 bash, gcc 等工具写了一个小小的核心程序，这个核心程序可以在 Intel 的 386 机器上面运作，让很多人很感兴趣！从此开始了 Linux 不平凡的路程！

关于 GNU 计划

1984 年创立 GNU 计划与 FSF 基金会的 Stallman 先生认为， 写程序最大的快乐就是让自己发展的良好的软件让大家来使用了！ 而既然程序是想要分享给大家使用的，不过，每个人所使用的计算机软硬件并不相同， 既然如此的话，那么该程序的原始码 (Source code) 就应该要同时释出， 这样才能方便大家修改而适用于每个人的计算机中呢！这个将原始码释出的举动， 就称为 Open Source ！此外，史托曼同时认为，如果您将您程序的 Source code 分享出来时，若该程序是很优秀的，那么将会有很多人使用， 而每个人对于该程序都可以查阅 source code ，无形之中，就会有一票人帮您除错啰！ 您的这支程序将会越来越壮大！越来越优秀呢！

而为了避免自己的开发出来的 Open source 的自由软件被拿去做成专利软件， 于是 Stallman 同时将 GNU 与 FSF 发展出来的软件，都挂上 GPL 的版权宣告～ 这个 FSF 的核心观念是『版权制度是促进社会进步的手段， 版权本身不是自然权力。』对于 FSF 有兴趣或者对于 GNU 想要更深入的了解时，请参考树德大学的洪朝贵教授的网站 http://saturn.stu.edu.tw/~ckhung/a/c_83.php，或直接到 GNU 去：http://www.gnu.org 里面有更为深入的解说！

Tips:
为什么要称为 GNU 呢？其实 GNU 是 GNU's Not Unix 的缩写，意思是说， GNU 并不是 Unix 啊！那么 GNU 又是什么呢？ 就是 GNU's Not Unix 嘛！.....如果您写过程序， 就会知道，这个 GNU = GNU's Not Unix 可是无穷循环啊！忙碌～



另外，什么是 Open Source 呢？所谓的 source 是程序发展者写出的原始程序代码， Open Source 就是，软件在发布时，同时将作者的原始码一起公布的意思！

那么这个 GPL (GNU General Public License, GPL) 是什么玩意儿？ 为什么要将自由软件挂上 GPL 的『版权宣告』呢？这个版权宣告对于作者有何好处？ 首先， Stallman 对 GPL 一直是强调 Free 的，这

个 Free 的意思是这样的：

"Free software" is a matter of liberty, not price. To understand the concept, you should think of "free speech", not "free beer". "Free software" refers to the users freedom to run, copy, distribute, study, change, and improve the software

大意是说， Free Software（自由软件）是一种自由的权力，并非是『价格！』 举例来说，你可以拥有自由呼吸的权力、你拥有自由发表言论的权力， 但是，这并不代表您可以到处喝『免费的啤酒！(free beer)』，也就是说， 自由软件的重点并不是指『免费』的，而是指具有『自由度，freedom』的软件， 史托曼进一步说明了自由度的意义是： 使用者可以自由的执行、复制、再发行、学习、修改与强化自由软件。 这无疑是个好消息！因为如此一来，你所拿到的软件可能原先只能在 Unix 上面跑， 但是经过原始码的修改之后，你将可以拿他在 Linux 或者是 Windows 上面来跑！总之，一个软件挂上了 GPL 版权宣告之后，他自然就成了自由软件！ 这个软件就具有底下的特色：

- 取得软件与原始码：您可以根据自己的需求来执行这个自由软件；
- 复制：您可以自由的复制该软件；
- 修改：您可以将取得的原始码进行程序修改工作，使之适合您的工作；
- 再发行：您可以将您修改过的程序，再度的自由发行，而不会与原先的撰写者冲突；
- 回馈：您应该将您修改过的程序代码回馈于社群！

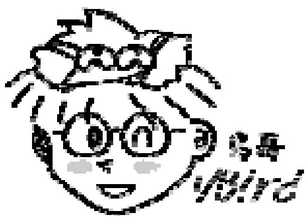
但请特别留意，您所修改的任何一个自由软件都不应该也不能这样：

- 修改授权：您不能将一个 GPL 授权的自由软件，在您修改后而将他取消 GPL 授权～
- 单纯贩卖：您不能单纯的贩卖自由软件。

也就是说，既然 GPL 是站在互助互利的角度上去开发的，您自然不应该将大家的成果占为己有， 而取消 GPL 授权的！对吧！因此您当然不可以将一个 GPL 软件的授权取消， 即使您已经对该软件进行大幅度的修改！那么自由软件也不能贩卖吗？当然不是！还记得上一个小节里面， 我们提到史托曼藉由贩卖 Emacs 取得一些经费，让自己生活不至于匮乏吧？？是的！ 自由软件是可以贩卖的，不过，不可仅贩卖该软件，应同时搭配售后服务与相关手册～ 这些可就需要工本费了呢！

很多人还是有疑问，目前不是有很多 Linux 开发商吗？为何他们可以贩卖 Linux 这个 GPL 授权的软件？原因很简单，因为他们大多都是贩卖『售后服务！』所以，他们所使用的自由软件， 都可以在他们的网站上面下载！（当然，每个厂商他们自己开发的工具软件就不是 GPL 的授权软件了！）但是，您可以购买他们的 Linux 光盘，如果您购买了光盘，他们会提供相关的手册说明文件， 同时也会提供您数年不等的咨询、售后服务、软件升级与其它协力工作等等的附加价值！ 所以说，目前自由软件工作者，他们所赖以维生的，几乎都是在『服务』这个领域呢！ 毕竟自由软件并不是每个人都会撰写，有人有需要您的自由软件时，他就会请求您的协助， 此时，您就可以透过服务来收费了！这样来说， 自由软件确实还是具有商业空间的喔！

Tips:
很多人对于 GPL 授权一直很疑惑，对于 GPL 的商业行为更是无法接受！ 关于这一点，鸟哥在这里还是要再次的申明， GPL 是可以从事商业行为的！ 而很多的作者也是藉由这些商业行为来得以取得生活所需，更进一步去发展更优秀的自由软件！ 千万不要听到



『商业』就排斥！这对于发展优良软件的朋友来说，是不礼貌的！

上面提到的大多是与使用者有关的项目，那么 GPL 对于自由软件的作者有何优点呢？ 大致的优点有这些：

- 软件安全性较佳；
- 软件执行效能较佳；
- 软件除错时间较短；
- 贡献的原始码远永都存在。

这是因为既然是 Open Source 的自由软件，那么您的程序代码将会有很多人帮您查阅， 如此一来，程序的漏洞与程序的优化将会进展的很快！所以，在安全性与效能上面， 自由软件一点都不输给商业软件喔！此外，因为 GPL 授权当中，修改者并不能修改授权， 因此，您如果曾经贡献过程序代码，嘿嘿！您将名留青史呢！不错吧！ ^_^

不过，就鸟哥的观点来看， GPL 对于程序开发者的优点是相当多的，不过， 对于不熟悉程序的一般人来说， GPL 的优点其实不太容易看出来～ 首先，虽然他是随手可得的自由软件，不过，您也必须要会使用基本的编译器才行吧！ (呵呵！这也是您为何要买这本书/察看鸟哥的网站的原因吧！ ^_^) 这对于一般人来说并不容易！当然啦，如果每个人都跟 Stallman 一样神，那商业公司就不用存在啦！ 嘿嘿！对于不懂程序的人来说，商业公司是一个很快速的解决之道啊！ 而对于我们广大的读者群来说，认识了/学习了 Linux 与自由软件的相关技巧后， 对于未来真的是有很不错的帮助喔！

Torvalds 的 Linux 发展

我们前面一节当中，提到了 Linux 是由 Torvalds 这个芬兰人所发明的。那么为何托瓦兹可以发明 Linux 呢？凭空想象而来的？还是有什么渊源？ 这里我们就来谈一谈啰！

与 Minix 之间

Linus Torvalds (托瓦兹, 1969 年出生)的外祖父是赫尔辛基大学的统计学家， 他的外祖父为了让自己的小孙子能够学点东西， 所以从小就将托瓦兹带到身边来管理一些微计算机。在这个时期，托瓦兹接触了汇编语言 (Assembly Language)，那是一种直接与芯片对谈的程序语言，也就是所谓的低级语言。 必须要很了解硬件的架构，否则很难以汇编语言撰写程序的。

在 1988 年间，托瓦兹顺利的进入了赫尔辛基大学，并选读了计算机科学系。 在就学期间，因为学业的需要与自己的兴趣，托瓦兹接触到了 Unix 这个操作系统。 当时整个赫尔辛基只有一部最新的 Unix 系统，同时仅提供 16 个终端机 (terminal)。 还记得我们上一节刚刚提过的，早期的计算机仅有主机具有运算功能，terminal 仅负责提供 Input/Output 而已。在这种情况下，实在很难满足托瓦兹的需求，因为..... 光是等待使用 Unix 的时间，就很耗时～ 为此，他不禁想到：『我何不自己搞一部 Unix 来玩？』不过，就如同 Stallman 当初的 GNU 计划一样，要写核心程序，谈何容易～

不过，幸运之神并未背离托瓦兹，因为不久之后，他就知道有一个类似 Unix 的系统， 并且与 Unix 完全兼容，还可以在 Intel 386 机器上面跑的操作系统， 那就是我们上一节提过的，谭宁邦教授为了教育需要而撰写的 Minix 系统！ 他在购买了最新的 Intel 386 的个人计算机后，就立即安装了 Minix 这个操作系统。 另外，由于 Minix 这个操作系统是有附上原始码的～所以，托瓦兹也经由这个原始码 学习到了很多的核心程序设计的设计概念喔！

对 386 硬件的多任务测试

事实上，托瓦兹对于个人计算机的 CPU 其实并不满意，因为他之前碰的计算机都是工作站型的计算机， 这类计算机的 CPU 特色就是可以进行『多任务处理』的能力。什么是多任务呢？ 理论上，一个 CPU 在一个时间内仅能进行一项工作， 那如果有两个工作同时出现到系统中呢？举例来说， 您可以在现今的计算机中同时开启两个以上的办公软件，例如电子电子表格与文字处理软件。 这个同时开启的动作代表着这两个工作同时要交给 CPU 来处理～ 啊！ CPU 一个时间点内仅能处理一个工作，那怎么办？没关系， 这个时候如果具有多任务的 CPU 就会自动在不同的工作间切换～ 亦即我先跑 10% 的电子电子表格，再转到文书处理器跑 10% ，再回去电子电子表格.... 一直到将两个工作结束为止(不一定同时结束！如果某个工作先结束了， CPU 就会全速去跑剩下的那个工作了！)

Tips:
为什么有的时候我同时开两个档案(假设为 A, B 档案)所花的时间， 要比开完 A 再去开 B 档案的时间还要多？现在是否稍微可以理解？ 因为如果同时开启的话，CPU 就必须要在两个工作之间不停的切换～ 而切换的动作还是会耗去一些 CPU 时间的！所以啰，同时启用两个以上的工作在一个 CPU 上， 要比一个一个的执行还要耗时一点。这也是为何现在 CPU 开发商要整合两个 CPU 于一个芯片中！ 也是为何在运作情况比较复杂的服务器上，需要比较多的 CPU 负责的原因！



早期 Intel 的 x86 架构计算机不是很受重视的原因，就是因为 x86 的芯片对于多任务的处理不佳， CPU 在不同的工作之间切换不是很顺畅。但是这个情况在 386 计算机推出后， 有很大的改善。托瓦兹在得知新的 386 芯片的相关信息后，他认为， 以价格性能比的观点来看， Intel 的 386 便宜而且性能上也就稍微可以将就将就 ^_^。 所以他就贷款去买了一部 Intel 的 386 来玩。

前面提到，托瓦兹是玩汇编语言的，汇编语言对于硬件有很密切的关系， 为了彻底发挥 386 的效能，于是托瓦兹花了不少时间在测试 386 机器上！ 他的重要测试就是在测试 386 的多功效能上。首先，他写了两个小程序， 一个程序会持续输出 A ，另一个会持续输出 B，他将两个程序同时执行， 结果，他看到屏幕上很顺利的一直出现 ABABABAB...... 他知道，他成功了！ ^_^

Tips:
要达到多任务(multitasking)的环境，除了硬件(主要是 CPU)需要能够具有多任务的特性外， 操作系统也需要支持这个功能喔！一些不具有多任务特性的操作系统， 想要同时执行两个程序是不可能的。除非先被执行的程序执行完毕，否则， 后面的程序不可能被主动执行。至于多任务的操作系统中，每个程序被执行时， 都会有一个最大 CPU 使用时间，若该工作运作的时间超过这个 CPU 使用时间时， 该工作就会先被丢出 CPU 的运作中，而再度的进入核心工作排程中，等待下一次的 CPU 运作。 这有点像在开记者会啦，主持人(CPU)会问『谁要发问』？一群记者(工作程序) 就会举手(看谁的工作重要！)，先举手的自然就被允许发问，问完之后，主持人又会问一次谁要发问，当然，所有人(包括刚刚那个记者)都可以举手！ 如此一次一次的将工作给他完成啊！ ^_^ 多任务的环境对于复杂的工作情况，帮助很大喔！



初次释出 Linux 0.02

探索完了 386 的硬件相关信息，并且也安装了类似 Unix 的 Minix 操作系统， 同时还取得 Minix 的原始码，接下来，托瓦兹干嘛去了？？ 因为 Minix 的发展控制在谭宁邦教授手上，他希望 Minix 能以教育的立场去发展，所以对于 Minix 的开发并不是十分的热中，但是一堆人对于 Minix 的功能需求又很强烈，例如一些接口与周边的驱动程序与新的协议等等。在无法快速的得到解决后， 托瓦兹就想，那我干脆自己写一个更适合我自己用的 Minix 好了！ 于是他就开始进行核心程序的撰写了。

对于托瓦兹来说，GNU 真的是一个不可多得的好家伙～ 因为他用来撰写属于自己小核心的工具，就是 GNU 的 bash 操作接口与 gcc 编译器等等自由软件。 他以 GNU 的软件针对 386 并参考 Minix 的设计理念(注意，仅是程序设计理念， 并没有使用 Minix 的原始码)来写这个小核心。喝！没想到竟然可以写出这个小玩意， 而这个小玩意竟然可以在 386 上面顺利的跑起来～还可以读取 Minix 的档案系统。 真是太好了！不过还不够，他希望这个程序可以获得大家的一些修改建议， 于是他便将这个核心放置在网络上提供大家下载，同时在 BBS 上面贴了一则消息：

```
Hello everybody out there using minix-
I'm doing a (free) operation system (just a hobby,
won't be big and professional like gnu) for 386(486) AT clones.
```

他说，他完成了一个好玩的小核心操作系统，这个核心是用在 386 机器上的， 同时，他真的仅是好玩，并不是想要做一个跟 GNU 一样大的计划！这则新闻引起很多人的注意， 他们也去托瓦兹提供的网站上下载了这个核心来安装。 有趣的是，因为托瓦兹放置核心的那个 FTP 网站的目录为： Linux ， 从此，大家便称这个核心为 Linux 了。(请注意，此时的 Linux 就是那个 kernel 喔！ 另外，托瓦兹所丢到该目录下的第一个核心版本为 0.02 呢！)

同时，为了让自己的 Linux 能够兼容于 Unix 系统，于是托瓦兹开始将一些能够在 Unix 上面运作的软件拿来在 Linux 上面跑。不过，他发现到，是有很多的软件无法在 Linux 这个核心上运作。 这个时候他有两种作法，一种是修改软件，让该软件可以在 Linux 上跑， 另一种则是修改 Linux ，让 Linux 符合软件能够运作的规范！ 由于 Linux 希望能够兼容于 Unix ，于是托瓦兹选择了第二个作法『修改 Linux 』！为了让所有的软件都可以在 Linux 上执行，于是托瓦兹开始参考标准的 POSIX 规范。

这个正确的决定让 Linux 在起步的时候体质就比别人优良～ 因为 POSIX 标准主要是针对 Unix 与一些软件运行时候的标准规范， 只要依据这些标准规范来设计的核心与软件，理论上，就可以搭配在一起执行了。 而 Linux 的发展就是依据这个 POSIX 的标准规范， Unix 上面的软件也是遵循这个规范来设计的，如此一来，让 Linux 很容易就与 Unix 兼容共享互有的软件了！ 同时，因为 Linux 直接放置在网络下，提供大家下载， 所以在流通的速度上相当的快！导致 Linux 的使用率大增！ 这些都是造成 Linux 大受欢迎的几个重要因素呢！

Linux 的发展：虚拟团队的产生

Linux 虽然是托瓦兹发明的，而且内容还绝不会涉及专利软件的版权问题。 不过，如果单靠托瓦兹自己一个人的话，那么 Linux 要茁壮实在很困难～ 因为一个人的力量是很有限的。好在托瓦兹选择 Linux 的开发方式相当的务实！ 首先，他将释出的 Linux 核心放置在 FTP 上面，并请告知大家新的版本信息， 等到使用者下载了这个核心并且安装之后，如果发生问题， 或者是由于特殊需求亟需某些硬件的驱动程序，那么这些使用者就会主动回报给托瓦兹。在托瓦兹能够解决的问题范围内，他都能很快速的进行 Linux 核

心的更新与除错。

不过，托瓦兹总是有些硬件无法取得的啊，那么他当然无法帮助进行驱动程序的撰写与相关软件的改良。这个时候，就会有些志工跳出来说：『这个硬件我有，我来帮忙写相关的驱动程序。』 因为 Linux 的核心是 Open Source 的，黑客志工们很容易就能够跟随 Linux 的原本设计架构， 并且写出兼容的驱动程序或者软件。志工们写完的驱动程序与软件托瓦兹是如何看待的呢？ 首先，他将该驱动程序/软件带入核心中，并且加以测试。 只要测试可以运行，并且没有什么主要的大问题，那么他就会很乐意的将志工们写的程序代码加入核心中！ 总之，托瓦兹是个很务实的人，对于 Linux 核心所欠缺的项目，他总是『先求有且能跑， 再求进一步改良』的心态！这让 Linux 使用者与志工得到相当大的鼓励！ 因为 Linux 的进步太快了！使用者要求虚拟内存，结果不到一个星期推出的新版 Linux 就有了！ 这不得不让人佩服啊！

另外，为因应这种程序代码的加入，于是 Linux 便逐渐发展成具有模块的功能！ 亦即是将某些功能独立出于核心外，在需要的时候才加载到核心中。如此一来， 如果有新的硬件驱动程序或者其它协议的程序代码进来时，就可以模块化， 大大的增加了 Linux 核心的可维护能力！

后来，因为 Linux 核心加入了太多的功能， 光靠托瓦兹一个人进行核心的实际测试并加入核心原始程序实在太费力～ 结果，就有很多的朋友跳出来帮忙这个前置作业！ 例如考克斯(Alan Cox)、与崔迪(Stephen Tweedie)等等， 这些重要的副手会先将来自志工们的修补程序或者新功能的程序代码进行测试， 并且结果上传给托瓦兹看，让托瓦兹作最后核心加入的原始码的选择与整并！ 这个分层负责的结果，让 Linux 的发展更加的容易！

特别值得注意的是，这些托瓦兹的 Linux 发展副手，以及自愿传送修补程序的黑客志工， 其实都没有见过面，而且彼此在地球的各个角落，大家群策群力的共同发展出现今的 Linux ， 我们称这群人为虚拟团队！而为了虚拟团队数据的传输，于是 Linux 便成立的核心网站： http://www.kernel.org ！ 而这群素未谋面的虚拟团队们，在 1994 年终于完成的 Linux 的核心正式版！ version 1.0 。 这一版同时还加入了 X Window System 的支持呢！更于 1996 年完成了 2.0 版， 同时因应商业版本的需求， 于是开始将核心版本以测试版及稳定版同时开发， 次版本偶数为稳定版，奇数为开发中的测试版。 例如 2.6 与 2.5 版为相同的版本，不过， 2.6 为稳定版， 2.5 则为测试版。 测试版含有较多的功能，不过，稳定性可不敢说～ 并且托瓦兹指明了企鹅为 Linux 的吉祥物。

> Tips:
> 奇怪的是，托瓦兹是因为小时候去动物园被企鹅咬了一口念念不忘， 而正式的 2.0 推出时，大家要他想一个吉祥物。他在想也想不到什么动物的情况下， 就将这个念念不忘的企鹅当成了 Linux 的吉祥物了......



Linux 由于托瓦兹是针对 386 写的，跟 386 硬件的相关性很强，所以， 早期的 Linux 确实是不具有移植性的。不过，大家知道 Open source 的好处就是， 可以修改程序代码去适合作业的环境。因此，在 1994 年以后， Linux 便被开发到很多的硬件上面去了！ 目前除了 x86 之外， IBM、HP、Sun 等等公司出的硬件也都有被 Linux 所支持呢！

Linux distributions

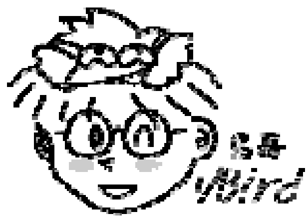
好了，经过上面的说明，我们知道了 Linux 其实就是一个操作系统最底层的核心及其提供的核心工具。 他是 GNU 授权模式，所以，任何人均可取得原始码与可执行这个核心程序，并且可以修改。此外，因为 Linux

参考 POSIX 设计规范，于是兼容于 Unix 操作系统， 故亦可称之为 Unix Like 的一种。

Linux 的出现让 GNU 计划放下了心里的一块大石头，因为 GNU 一直以来就是缺乏了核心程序， 导致他们的 GNU 自由软件只能在其它的 Unix 上面跑。既然目前有 Linux 出现了，且 Linux 也用了很多的 GNU 相关软件，所以 Stallman 认为 Linux 的全名应该称之为 GNU/Linux 呢！ 不管怎么说，Linux 实在很不错，让 GNU 软件大多以 Linux 为主要操作系统来进行开发，此外，很多其它的自由软件团队，例如 sendmail, wu-ftp, apache 等等也都有以 Linux 为开发测试平台的计划出现！如此一来， Linux 除了主要的核心程序外，可以在 Linux 上面运行的软件也越来越多，如果有心，就能够将一个完整的 Linux 操作系统搞定了！

虽然由 Torvalds 负责开发的 Linux 仅具有 Kernel 与 Kernel 提供的工具， 不过，如上所述，很多的软件已经可以在 Linux 上面运作了，因此，Linux + 各家软件就可以完成一个相当完整的操作系统了。不过，要完成这样的操作系统......还真难～ 因为 Linux 早期都是由黑客工程师所开发维护的，他们并没有考虑到一般使用者的能力...... 为了让使用者能够接触到 Linux ，于是很多的商业公司或非营利团体，就将 Linux Kernel (含 tools) 与可运行的软件整合起来，加上自己具有创意的工具程序， 这个工具程序可以让使用者以光盘或者透过网络直接安装/管理 Linux 系统。 这个 Kernel + Softwares + Tools 的可完整安装的咚咚，我们称之为 Linux distribution ， 一般中文翻译成 可完整安装套件，或者安装套件等等。

Tips:
由于 Linux 核心是由黑客工程师写的，要由原始码安装到 x86 计算机上面成为可以执行的 binary 档案， 这个过程可不是人人都会的～所以早期确实只有工程师对 Linux 有兴趣。一直到一些社群与商业公司将 Linux 核心配合自由软件，并提供完整的安装程序， 且制成光盘后，对于一般使用者来说， Linux 才越来越具有吸引力！ 因为只要一直『下一步』就可以将 Linux 安装完成啊！ ^_^



我们前面说过， GNU 的 GPL 授权并非不能从事商业行为，于是很多商业公司便成立来贩卖 Linux distribution。而由于 Linux 的 GPL 版权宣告，因此，商业公司所贩卖的 Linux distributions 通常也都可以从 Internet 上面来下载的！此外，如果您想要其它商业公司的服务， 那么直接向该公司购买光盘来安装，也是一个很不错的方式的！

不过，由于发展 Linux distributions 的公司实在太多了，例如有名的 Red Hat, Mandriva, Debian, SuSE 等等，所以很多人都很担心，如此一来每个 distribution 是否都不相同呢？这就不需要担心了，因为每个 Linux distributions 使用的 kernel 都是 http://www.kernel.org 所释出的，而他们所选择的软件，几乎都是目前很知名的软件，重复性相当的高， 例如 WWW 服务器的 Apache ， Mail 服务器的 Postfix/sendmail ，File 服务器的 Samba 等等。

此外，为了让所有的 Linux distributions 开发不致于差异太大，还有 Linux Standard Base (LSB) 来规范开发者，以及目录架构的 File system Hierarchy Standard (FHS) 规范！ 唯一差别的，可能就是该开发者自家所开发出来的管理工具，以及套件管理的模式吧！ 所以说，基本上，每个 Linux distributions 除了架构的严谨度与选择的套件内容外， 其实差异并不太大啦！ ^_^ 。大家可以选择自己喜好的 distribution 来安装即可！

底下列出几个主要的 Linux distributions 发行者网址：

- Red Hat: http://www.redhat.com
- Fedora: http://fedora.redhat.com
- Mandriva: http://www.mandriva.com
- Novell SuSE: http://www.novell.com/linux/suse/
- Debian: http://www.debian.org/
- Slackware: http://www.slackware.com/
- Linpus: http://www.linpus.com.tw/
- Gentoo: http://www.gentoo.org/
- Ubuntu: http://www.ubuntulinux.org/
- CentOS: http://www.centos.org/

当然发行套件者不仅于此，您可以查阅其它的 Linux 新闻来发现喔！ 但是值得大书特书的，是中文 Linux 的延伸计划： CLE 这个套件！ 早期的 Linux 因为是工程师发展的，而这些工程师大多以英文语系的国家为主， 所以， Linux 对于国人的学习是比较困扰一点。后来由国人发起的 CLE 计划：http://cle.linux.org.tw/ 开发很多的中文套件级翻译了很多的英文文件，使得我们目前得以使用中文的 Linux 呢！ 另外，目前正在开发中的还有台南县卧龙小三等老师们发起的众多自由软件计划， 真是造福很多的朋友啊！

- 自由软件技术交流网： http://freesf.tnc.edu.tw/index.php
- B2D: http://b2d.tnc.edu.tw/

此外，如果只想看看 Linux 的话，还可以选择所谓的可光盘开机进入 Linux 的 Live CD 版本， 亦即是 KNOPPIX 这个 Linux distributions 呢！台湾也有阿里巴巴兄维护的中文 Live CD 喔！

- http://www.knoppix.net/
- 中文 KNOPPIX: http://knoppix.tnc.edu.tw/

Tips:
对于没有额外的硬盘或者是没有额外的主机的朋友来说， KNOPPIX 这个可以利用光盘开机而进入 Linux 操作系统的 Live CD 真的是一个不错的选择！您只要下载了 KNOPPIX 的映象档，然后将他烧录成为 CD，放入您主机的光驱，并设定光盘为第一个开机选项，就可以使用 Linux 系统了呢！



如果您还想要知道更多的 Linux distributions 的下载与使用信息，可以参考：

- http://www.linuxiso.org/
- http://distrowatch.com/

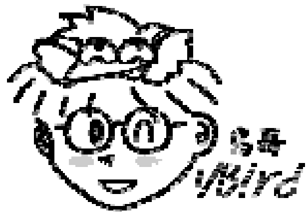
那我到底应该要选择哪一个 distributions ？就如同我们上面提到的，其实每个 distributions 差异性并不大！ 不过，由于套件管理的方式主要分为 Debian 的 pkg 及 Red Hat 系统的 RPM 方式， 目前鸟哥的建议是，先学习以 RPM 套件管理为主的 Fedora/SuSE/Mandriva 等台湾使用者较多的版本， 这样一来，发生问题时，可以提供解决的管道比较多。如果您已经接触过 Linux 了， 还想要更严谨的 Linux 版本，那可以考虑使用 Debian ，如果您是以效能至上来考虑， 那么或许 Gentoo 是不错的建议！ 总之，版本很多，但是各版本差异其实不大，建议您一定要先选定一个版本后， 先彻头彻尾的了解他，那再继续玩其

它的版本时，就可以很快的进入状况。 鸟哥的书/网站仅提供一个版本，不过是以比较基础的方式来介绍的， 因此，如果能够熟练这本书/网站的话，呵呵！哪一个 distributions 对您来说，都不成问题啦！

Linux 的特色

Linux 是 Torvalds 先生所开发出来的，基于 GPL 的版权宣告之下， 可以在 x86 的架构下运作，也可以被移植到其它的大型主机上面。 由于开发的相关理念与兼容的问题，因此，我们也可以称 Linux 为 Unix Like 操作系统的一种。

> Tips:
> 其实 Unix-Like 可以说是目前服务器类型的操作系统的统称啦！ 因为，不论是 FreeBSD, BSD, Sun Unix, HP Unix, Red Hat Linux, Mandrake Linux 等等，都是由同一个祖先 Thompson 所写的『Unix』来的， 因此，这些咚咚都被统称为 Unix-Like 的操作系统啰！



Linux 的特色

那么这个系统有什么特异功能呢？简单的说：

- 自由与开放：由于 Linux 是基于 GPL (General Public License) 的架构之下，因此他是自由软件， 也就是任何人都可以自由的使用或者是修改其中的原始码的意思！ 这就是所谓的『开放性架构』， 这对科学界来说是相当重要的！因为很多的工程师由于特殊的需求，常常需要修改系统的原始码， 使该系统可以符合自己的需求！而这个开放性的架构将可以满足各不同需求的工程师！ 因此当然就有可能越来越流行啰！以鸟哥来说，目前环境工程界的空气质量模式最新版 Models-3/CMAQ 就是以 Linux 为基准平台设计的呢！

- 配备需求低廉：而 Linux 可以支持个人计算机的 x86 架构， 系统资源不必像早先的 Unix 系统那般，仅适合于单一公司 (例如 Sun) 设备！ 单就这一点来看，就可以造成很大的流行啰！不过，如果您想要在 Linux 下执行 X Window 系统， 那么硬件的等级就不能太低了！

- 功能强大而稳定：而且由于 Linux 功能并不会输给一些大型的 Unix 工作站，因此，近年来越来越多的公司或者是团体、 个人投入这一个操作系统的开发与整合工作！

- 独立作业：另外， 由于很多的软件套件逐渐被这套操作系统拿来使用，而很多套件软件也都在 Linux 这个操作系统上面进行发展与测试，因此， Linux 近来已经可以独力完成几乎所有的工作站或服务器的服务了，例如 Web, Mail, Proxy, FTP.....。

所以，目前 Linux 已经是相当成熟的一套操作系统啰！而且不耗资源又可以自由取得！呵呵， 可以说造成微软相当大的压力呀！此外，由于他的系统硬件要求很低， 加上目前很多的人由于『Intel 的阴谋』(呵呵！开玩笑的，因为 Tom 的硬件评论 (http://www.big5.tomshardware.com/) 网站常常这样取笑 Intel 的说！呵！很好笑！) 而造成手边有相当多的淘汰掉的硬件配备， Linux 在这些被淘汰的硬件中就可以执行的相当的顺畅与稳定！ 因此也造成相当多朋友的关注啰！

这也是造成 Linux 成为最近几年来最受瞩目的操作系统之一，如前所述， 他会受到瞩目的原因主要是因为他是『free』的，就是可以自由取得的操作系统啦！ 然后他是开放性的系统，也就是你可以随时的取得

程序的原始码，这对于程序开发工程师是很重要的！ 而且，虽然他是 Free 的自由软件，不过功能却很强大！另外， Linux 对于硬件的需求是很低的，这一点更造成它流行的主因，因为硬件的汰换率太快了， 所以很多人手边都有一些很少在用的零件，这些零件组一组就可以用来跑 Linux 了，反正做一个工作站又不用使用到屏幕 （ 只要主机就可以啰 ），因此 Linux 就越来越流行啰！(插个嘴，也就是因为 Linux 具有 1.硬件需求低、 2.架构开放、 3.系统稳定性及保密性功能够强、 4.完全免费，所以造成一些所谓『反微软联盟』的程序设计高手不断的开发新软件！以与 Microsoft 进行抗衡！）

Linux 的优缺点

那干嘛要使用 Linux 做为我们的主机系统呢？这是因为 Linux 有底下这些优点：

- 稳定的系统：
  Linux 本来就是基于 Unix 概念而发展出来的操作系统，因此， Linux 具有与 Unix 系统相似的的程序接口跟操作方式，当然也继承了 Unix 稳定并且有效率的特点。常听到安装 Linux 的主机连续运做一年以上而不曾当机、不必关机是稀松平常的事；

- 免费或少许费用：
  由于 Linux 是基于 GPL 的基础下的产物，因此任何人皆可以自由取得 Linux ， 至于一些『安装套件』的发行者， 他们发行的安装光盘也仅需要些许费用即可获得！不同于 Unix 需要负担庞大的版权费用，当然也不同于微软需要一而再、 再而三的更新你的系统，并且缴纳大量费用啰！

- 安全性、漏洞的快速修补：
  如果你常玩网络的话，那么你最常听到的应该是 『没有绝对安全的主机』！没错！不过 Linux 由于支持者日众，有相当多的热心团体、个人参与其中的开发， 因此可以随时获得最新的安全信息，并给予随时的更新，亦即是具有相对的较安全！

- 多任务、多使用者：
  与 Windows 系统不同的， Linux 主机上可以同时允许多人上线来工作， 并且资源的分配较为公平，比起 Windows 的单人假多任务系统要稳定的多啰！ 这个多人多任务可是 Unix-Like 上面相当好的一个功能，怎么说呢？你可以在一部 Linux 主机上面规划出不同等级的使用者，而且每个使用者登入系统时的工作环境都可以不相同， 此外，还可以允许不同的使用者在同一个时间登入主机，以同时使用主机的资源。

- 使用者与群组的规划：
  在 Linux 的机器中，档案的属性可以分为『可读、可写、可执行』 等参数来定义一个档案的适用性，此外，这些属性还可以分为三个种类， 分别是『档案拥有者、档案所属群组、其它非拥有者与群组者』。 这对于项目计划或者其它计划开发者具有相当良好的系统保密性。

- 相对比较不耗资源的系统：
  Linux 只要一部 p-100 以上等级的计算机就可以安装并且使用愉快啰！还不需要到 P-4 或 AMD K8 等级的计算机呢！不过，如果你要架设的是属于大型的主机（服务上百人以上的主机系统），那么就需要比较好一点的机器了。不过，目前市面上任何一款个人计算机均可以达到这一个要求啰！

- 适合需要小核心程序的嵌入式系统：
  由于 Linux 只要几百 K 不到的程序代码就可以完成一个完整的操作系统，因此相当适合于目前家电或者是小电子用品的操作系统呢！那就是当红炸子鸡『嵌入式』系统啦！ Linux 真的是很适合例如手机、数字相机、PDA、家电用品等等的微电脑操作系统呢！ ^_^

反正 Linux 好处说不完啦！不过虽然 Linux 具有这样多的好处， 但是他先天上有一个足以致命的地方，使他的普及率受到很大的限制，就是 Linux 需要使用『指令列』 的终端机模式进行系统的管理！虽然近年来有很多的图形接口开发使用在 Linux 上面，但毕竟要熟悉 Linux 还是以指令列来使用是比较好的，因此要接受 Linux 的玩家必须比较要能熟悉对计算机下指令的行为，而不是用鼠标点一点 icon 就行了！不过如果只是要架一些简单的小站呢？是不是大家都可以做的到？没错！其实只要对 Linux 做一些小小的设定就可以架站了！Linux 还可以改进的地方：

- 没有特定的支持厂商：因为在 Linux 上面的所有套件几乎都是自由软件， 而每个自由软件的开发者可能并不是公司团体，而是非营利性质的团体。如此一来， 在您 Linux 主机上面的软件若发生问题，该如何是好？ 好在由于目前 Linux 商业界的整合还不错，目前在台湾比较具名的 Red Hat 与 SuSE 均有设立了服务点。您可以经由这个服务点来直接向他们购买/咨询相关的软硬件问题呢！ 不过，如果您并非选择有专门商业公司的 Linux distributions 时？怎么办？ 没有专人到府服务呢～这点倒是还不需要太担心， 因为拜网络风行之赐，你要问的问题几乎在网络上都可以找到答案喔！看你有没有用心去找就是了！

- 图形接口作的还不够好：虽然早在 1994 年 Linux 1.0 版释出时， 就已经含有 XFree86 的 X Window 架构了。不过， X Window 毕竟是 Linux 上的一个软件， 他并不是 Linux 最核心的部分，有没有他对 Linux 的服务器执行都没有影响的！ 所以鸟哥通常是不玩 X Window 的啦！ 但其实有更多人对于 Linux 并非是着眼在网络服务器，而是一般桌上型计算机的使用， 这一点对于 Linux 来说，还是不够好！即使目前已有 KDE (http://www.kde.org/) 及 GNOME (http://www.gnome.org/) 等优秀的窗口管理程序， 不过，毕竟整合度还是需要加强，希望未来可以看到整合度超高的 Linux 桌上型计算机呢！

其它 Linux 相关

还有一些数据需要提醒大家的呢！

- 依循标准：

Linux 有个优良的传统，那就是支持比较公认而正式的标准， 例如开发时就使用的 POSIX 规范。此外， 由于 Linux 是一个没有『规格品』样式的新鲜玩意儿， 如果大家都自己玩自己的，那么可想而知的是，未来想要在 Linux 这个操作系统上面发展软件与硬件的厂商一定会无所适从的！ 为了让软件开发商、与硬件发展者有一个依循的方向，因此而有 Linux Standard Base (LSB) 及 File system Hierarchy Standard (FHS) 这些个玩意儿的诞生！所以，各个 distribution 也都要遵循 LSB 上面的规范，软硬件发展者也都会依循 LSB ，所以啰，我们才会常常说，各大 distribution 虽然在提供的工具与创意上面有所不同，但是基本上，他们的 Linux 架构都是很类似的！因此，你只要玩一套 Linux distribution 也就够了！其它的 distribution 应该就难不倒你啰！

- FHS: http://www.pathname.com/fhs/

- LSB: http://www.linuxbase.org/

- 服务器、工作站、终端机...

由于网络的盛行，我们或许常常会听到一些名词ㄋㄟ～所以，底下先来介绍几个简单的网络名词吧！ ^_^，来认识一下什么是 服务器 (Server)、工作站 (Workstation) 与终端机 (Ternianl)？简单的来说，你可以这样认为：

- 服务器 (Server)：提供 Internet 一种以上的网络服务的主机，例如 yahoo 提供的是 WWW 的服务，那么 yahoo 就可以称之为 服务器 了！那么，你自己架设一台 mail server 呢？呵呵！那也是一个小型的服务器啰！所以，你必须要清楚的知道， 服务器是有规模大小之分的！目前一部 Linux 上面就可以架设多个服务器软件啰！例如 WWW 服务器的 Apache 软件、 FTP 服务器的 Wu-ftp 软件等等的；
- 工作站 (Workstation) ：基本上， 工作站可以视为仅提供一群特定人士，作为数值分析、科学用途的机器。例如鸟哥的研究室有一部 Sun 的机器，他仅提供我们研究室内部几部计算机的联机，当我们有需要使用到 Fortran 这个程序语言时，就联机进入 Sun 这部机器，并在上面进行我们所需要的计算工作！ 这个就是工作站啦。而工作站与服务器的差别，大概就在于有没有提供 internet 上面的服务而已，例如，如果我将 Sun 上面的 mail server 开启之后， 那么这部机器就可以称之为服务器了！同时也是我们的工作站喔！ 当然，更广义的定义是，只要是没有对 Internet 上面提供网络服务的，那就是工作站了！这当然也就包含所谓的终端机！
- 终端机 (Terminal) ：简单的说，就是 end-user (就是你啦) 前面的那部计算机！呵呵，例如鸟哥都是使用我的工作机 (Windows) 连上我们的主机来工作，那么这一部 Windows 的计算机，就可以称为 terminal 机器啰！不过，更狭义的来说，『终端机』本身应该是不具备任何可以作业的软件的， 在终端机上面一定要连上 Server 之后，才能进行各项作业！ 那才是最狭义的终端机啰！例如我们前面说过的 早期的大型主机联机模式！

大致上的分别可以用上面的观点来看！

- 几个常见的授权模式与定义

现在市面上有好多的软件，有的是自由软件，有的是专利软件。 有的专利软件免费，有的自由软件要钱～啊！好烦啊！怎么分辨这些东西？ 其实，鸟哥并不是律师，对于法律也不十分懂，不过， 还是有几个授权模式可以来谈一谈～

- Open Source ：Open source 表示软件释出时， 一定伴随着原始码的释出喔！通常他有几个好处：

    1. 程序设计师通常会等到程序成熟之后才会释出 (免得被笑, ^_^)， 所以通常程序在雏形的时候，就已经具有相当的优良体质；
    2. Open Source 的精神，相信当程序原设计人将程序原始码释出之后， 其它的程序设计师接受这份原始码之后，由于需要将程序改成自己所需的样式， 所以会经由本身的所学来加以改良，并从中加以改良与除虫，所以程序的 debug 功能会比传统的 close source 来的快！
    3. 由于程序是伴随原始码的，因此，系统将会不易存在鲜为人知的木马程序或一些安全漏洞， 相对而言，会比较更加的安全！

Open source 的代表授权为 GNU 的 GPL 及 BSD 等等：

- GNU General Public License ： 这个可以由刚刚的说明了解，他主要定义在『自由软件』上面，任何挂上 GPL 授权的软件，需要公布其原始码 (Open Source)，GPL 有几个主要的大方向：
    1. 任何个人或公司均可释出自由软件 (free software)；
    2. 任何释出自由软件的个人或公司，均可由自己的服务来收取适当的费用；
    3. 该软件的原始码 (Source Code) 需要随软件附上，并且是可公开发表的；
    4. 任何人均可透过任何正常管道取得此一自由软件，且均可取得此一授权模式。

  更多的 GPL 可参考附录 A 的中文化条文！

- Berkeley Software Distribution (BSD) ：使用 BSD source code 最常接触到的就是 BSD 授权模式了！这个授权模式其实与 GPL 很类似，而其精神也与 Open Source 相呼应呢！

此外，近期以来还有 Apache 的授权、 Sun 的授权，好多好多～

- Close Source ： 程序的核心是封闭的，优点是有专人维护，您不需要去更动他；缺点则是灵活度大打折扣， 使用者无法变更该程序成为自己想要的样式，此外，若有木马程序或者安全漏洞，将会花上相当长的一段时间来除错!这也是所谓专利软件 (copyright) 常见的软件出售方式。代表的授权模式有：
    - Freeware ：不同于 Free software ， Freeware 为『免费软件』而非『自由软件！』虽然它是免费的软件，但是不见得要公布其原始码， 端看释出者的意见啰！这个东西与 Open Source 毕竟是不太相同的东西喔！ 此外，目前很多标榜 免费软件 的程序很多都有小问题！例如假借免费软件的名义， 实施使用者数据窃取的目的！所以 『来路不明的软件请勿安装！』
    - Shareware ：共享软件这个名词就有趣了！ 与免费软件有点类似的是， Shareware 在使用初期，它也是免费的，但是， 到了所谓的『试用期限』之后，您就必须要选择『付费后继续使用』或者『将它移除』 的宿命～通常，这些共享软件都会自行撰写失效程序，让你在试用期限之后就无法使用该软件。

重点回顾

- 计算机主要以二进制作为单位，而目前常用的磁盘容量单位为 bytes，其单位换算为 1Byte = 8bits， 其它的以 1024 为其倍数，如 1GByte=1024MBytes 等等。
- 操作系统(Operation System)主要在管理与驱动硬件，因此必须要能够管理内存、管理装置、 负责行程管理以及系统呼叫等等。因此，只要能够让硬件准备妥当 (Ready) 的情况， 就是一个阳春的操作系统了。

- 最阳春的操作系统仅在驱动与管理硬件，而要使用硬件时，就得需要透过应用软件或者是壳程序(shell) 的功能，来呼叫操作系统操纵硬件工作。因此，目前称为操作系统的， 除了上述功能外，通常已经包含了日常工作所需要的应用软件在内了。
- Unix 的前身是由贝尔实验室(Bell lab.)的 Ken Thompson 利用组译语言写成的， 后来在 1971-1973 年间由 Dennis Ritchie 以 C 程序语言进行改写，才称为 Unix。
- 1977 年由 Bill Joy 释出 BSD (Berkeley Software Distribution)，这些称为 Unix-like 的操作系统。
- 1984 年由 Andrew Tannenbaum 制作出 Minix 操作系统，该系统可以提供原始码以及软件；
- 1984 年由 Richard Stallman 提倡 GNU 计划，倡导自由软件(Free software)， 强调其软件可以『自由的取得、复制、修改与再发行』，并规范出 GPL 授权模式， 任何 GPL (General Public License) 软件均不可单纯仅贩卖其软件，也不可修改软件授权。
- 1991 年由芬兰人 Linus Torvalds 开发出 Linux 操作系统。简而言之，Linux 成功的地方主要在于： Minix(Unix), GNU, Internet, POSIX 及虚拟团队的产生。
- Linux 本身就是个最阳春的操作系统，其开发网站设立在 http://www.kernel.org，我们亦称 Linux 操作系统最底层的数据为『核心, Kernel』。
- 目前 Linux 的发展分为两种版本，分别是稳定版本的偶数版，如 2.6.X，适合于商业与家用环境使用； 一种是发展中版本如 2.5.X 版，适合开发特殊功能的环境。
- Linux distributions 为 Linux 的 Kernel + Tools + Free Software + Documentations + 可完整安装的程序所制成的一套完整的系统。

习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 一个完整的操作系统至少要能够完整的控制整个硬件，请问，操作系统应该要控制硬件的哪些单元？

  至少要能够控制：(1)input/output crontrol, (2)device control, (3)process management, (4)file management. 等等!

- 核心的功能在于管控整个系统的硬件，这包括了 CPU 运算单元的管理，输入/输出的管理，内存的管理等等。 那么请问一个较为完整的操作系统，应该包含哪些部分？

  应包含 Kernel + Kernel Tools + Applications 等等

- 一个 GBytes 的硬盘空间，等于几个 KBytes ?

  1GBytes = 1024 MBytes * 1024 KBytes/MBytes = 1048576 KBytes

- 你在你的主机上面安装了一张网络卡，但是开机之后，系统却无法使用，您确定网络卡是好的， 那么可能的问题出在哪里？该如何解决？

因为所有的硬件都没有问题，所以，可能出问题的地方在于系统的核心(kernel) 不支持这张网络卡。解决的方法，到网络卡的开发商网站，下载支持您主机操作系统的驱动程序， 安装网络卡驱动程序后，就可以使用了。

- 我在一部主机上面安装 Windows 操作系统时，并且安装了显示卡的驱动程序，他是没有问题的。但是安装 Linux 时，却无法完整的显示整个 X Window 。请问，我可不可以将 Windows 上面的 显示卡驱动程序拿来安装在 Linux 上？

  不行！因为核心不同，针对硬件所写的驱动程序也会不相同， 编译器也不同，当然，驱动程序也无法在两个操作系统间兼容。 这也是为何开发商在他们的网站上面，都会同时提供许多不同操作系统的驱动程序之故。

- 我在 Windows 上面玩的游戏，可不可以拿到 Linux 去玩？

  当然不行！跟上一题相似的，因为游戏也是一个应用程序 (application)， 他必须要使用到核心所提供的工具来开发他的游戏， 所以这个游戏是不可在不同的平台间运作的。除非这个游戏已经进行了移植。

- 什么是软件的移植？

  程序是由程序代码(通成为 ASCII 格式)经过编译器编译成为 binary file 之后， 才能够在该操作系统上面执行。因此，您可以将您的程序的程序代码修改成可以适应 其它操作系统的环境，并且加以编译，使程序可以在其它平台上运作， 这个修改的动作即为移植。

- Linux 本身仅是一个核心与相关的核心工具而已，不过，他已经可以驱动所有的硬件， 所以，可以算是一个很阳春的操作系统了。经过其它应用程序的开发之后，被整合成为 Linux distribitions。请问众多的 distributions 之间，有何异同？

  相同：利用同样的 Linux kernel (http://www.kernel.org)，几乎相同的自由软件套件 (例如 GNU 里面的 gcc/glibc/vi/apache/bind/sendmail...)，几乎相同的操作接口 (例如均使用 bash/KDE/GNOME 等等)。
  不同：使用的 kernel 与各软件的版本可能会不同；各开发商加入的应用工具不同， 使用的套件管理模式不同(debian 与 RPM)

- Unix 是谁写出来的？ GNU 计划是谁发起的？

  Unix 是 Ken Thompson 写的，1973 年再由 Dennis Ritchie 以 C 语言改写成功。 至于 GNU 与 FSF 则是 Richard Stallman 发起的。

- GNU 的全名为何？他主要由那个基金会支持？

  GNU 是 GNU is Not Unix 的简写，是个无穷循环！ 另外，这个计划是由自由软件基金会 (Free Software Foundation, FSF) 所支持的！ 两者都是由 Stallman 先生所发起的！

- 我要如何取得 Linux distribution 的可安装光盘？

目前各大版本的 Linux distribution 大致上仍然依附在 GPL 这个版权宣告上， 因此在网络上面都可以轻易的下载，若有兴趣的话可以到各发展的厂商主网页去下载。 然而为了频宽的节省起见，建议在台湾以映像站台来下载速度上会快上很多， 特别建议南台湾朋友可以到义守大学的 FTP 站 (http://ftp.isu.edu.tw/)下载资料!

- 简单的说明一下什么是 GNU 的 GPL ?

1. 任何个人或公司均可释出自由软件 (free software);
2. 任何释出自由软件的个人或公司，均可由自己的服务来收取适当的费用;
3. 该软件的原始码 (Source Code) 需要随软件附上，并且是可公开发表的;
4. 任何人均可透过任何正常管道取得此一自由软件，且均可取得此一授权模式。

- 何谓多人 (Multi-user) 多任务 (Multitask)?

Multiuser 指的是 Linux 允许多人同时连上主机之外， 每个使用者皆有其各人的使用环境，并且可以同时使用系统的资源!
Multitask 指的是多任务环境，在 Linux 系统下， CPU 与其它例如网络资源可以同时进行多项工作， Linux 最大的特色之一即在于其多任务时，资源分配较为平均!

- 简单说明 GNU General Public License (GPL) 与 Open Source 的精神:

1. GPL 的授权之软件，乃为自由软件（Free software），任何人皆可拥有他； 2. 开发 GPL 的团体(或商业企业)可以经由该软件的服务来取得服务的费用； 3. 经过 GPL 授权的软件，其属于 Open source 的情况，所以应该公布其原始码； 4. 任何人皆可修改经由 GPL 授权过的软件，使符合自己的需求； 5. 经过修改过后 Open source 应该回馈给 Linux 社群。

- 有个朋友问我说『 Linux 是什么? 』我该如何回答比较好?

简单的说， Linux 就是一个操作系统，或者说， Linux 是操作系统最底层的核心。 这个核心可以管理整个计算机硬件，让计算机硬件可以完整的运作起来，并等待使用者输入指令。最早 Linux 是由 Torvalds 在 1991 年写出来的，后来由于他承接了 Unix 的良好传统：稳定性高、 多人多任务的环境设计优良、要求配备较低等优点，所以很多软件开发商在这个核心上面开发， 而某些厂商将这些软件与核心整合成为可以完整安装的光盘，而成为目前大家常常听到的 Linux 操作系统了。

- 什么是 POSIX ?为何说 Linux 使用 POSIX 对于发展有很好的影响?

POSIX 是一种标准规范，主要针对在 Unix 操作系统上面跑的程序来进行规范。 若您的操作系统符合 POSIX ，则符合 POSIX 的程序就可以在您的操作系统上面运作。 Linux 由于支持 POSIX ，因此很多 Unix 上的程序可以直接在 Linux 上运作， 因此程序的移植相当简易! 也让大家容易转换平台，提升 Linux 的使用率。

- Linux 的发展主要分为哪两种核心版本?

主要分为奇数的发展中版本(develop)，如 2.5 ，及偶数的稳定版本，如 2.6 。

- 简单说明自由软件 (free software) 与开放源码 (open source) 的差异?

  自由软件意指：你可以拥有自由的取得、复制、修改、再发行该软件的权利， 由于具有这些权利，因此自由软件通常是 Open source 的。
  开放源码意指：软件释出时，同时释出原始码，但使用者取得原始码后，能否修改该原始码， 则依据该软件的授权而定。
  意思就是说，自由软件是 Open source 的，但是 Open source 的软件则不见得是自由软件!

- 什么是 Linux 的 Live CD ?

  所谓的 Live CD 就是将完整的 Linux distribution 放置到一片光盘 (目前也有 DVD 版本了) 当中， 然后透过重新开机以『光盘开机』，就能够不使用硬盘直接进入 Linux 系统的环境。

- 简单说明 Linux 成功的因素?

  1. 藉由 Minix 操作系统开发的 Unix like ，没有版权的纠纷;
  2. 藉助于 GNU 计划所提供的各项工具软件， gcc/bash 等;
  3. 藉由 Internet 广为流传;
  4. 藉由支持 POSIX 标准，让核心能够适合所有软件的开发;
  5. 托瓦兹强调务实，虚拟团队的自然形成!

参考数据

- 王孝熙等，2002，『计算器概论』，台湾东华书局出版。
- 葛林穆迪着，杜默译，『Linux 传奇』，时报文化出版企业。
- 网络农夫，2001， Unix 简史
  http://netlab.cse.yzu.edu.tw/~statue/freebsd/docs/csh/
- Ken Thompson 的个人网站： http://plan9.bell-labs.com/cm/cs/who/ken/index.html
- Dennis Ritchie 的个人网站： http://cm.bell-labs.com/cm/cs/who/dmr/
- Richard Stallman 的个人网站： http://www.stallman.org/
- GNU 计划： http://www.gnu.org
- 洪朝贵老师的 GNU/FSF 介绍： http://saturn.stu.edu.tw/~ckhung/a/c_83.php

---

如何学习 Linux

最近更新日期：2005/06/08

目前 Linux 上头有两种主要的操作模式，分别是图形接口与文字接口， 那么学习 Linux 要用 X-Window (图形接口) 好还是 Command Line (文字接口) 好？ 这两种学习心态有什么优缺点呢？此外，有没有良好的入门文件可供参考？！ Linux 学习有困扰的时候应该要如何发问？！要到哪里去搜寻网络资源？！还有，怎样进行有智慧的提问？嗯！在这一章里面，我们好好谈一谈！

Linux 的应用

前一章节我们谈到了 Linux 相关的历史， 与简单的介绍了一下 Linux 这个『Kernel』与 Linux distributions 等等。 而在开始进入 Linux 的基础学习之前，我们有必要了解一下应该要如何有效的学习 Linux 的！ 但在谈到 Linux 如何学习之前，我们得就 Linux 目前的一般应用来说明一下， 才好理解您需要什么样的学习方式！目前 Linux 的应用至少有底下这些：

- 网络服务器：

承袭了 Unix 高稳定性的良好传统， Linux 上面的网络功能特别的稳定与强大！ 此外，由于 GNU 计划与 Linux 的 GPL 授权模式，让很多优秀的软件都在 Linux 上面发展， 且这些在 Linux 上面的服务器软件几乎都是自由软件！ 因此，做为一部网络服务器，例如 WWW, Mail Server, File Server, FTP Server 等等， Linux 绝对是上上之选！当然，这也是 Linux 的强项！

- 工作站计算机：

如同前一章提到的，工作站计算机与服务器不一样的地方，大概就是在于网络服务了。 工作站计算机本身是不应该提供 Internet 的服务的(LAN 内的服务则可接受)。 此外，工作站计算机与桌上型计算机不太一样的地方，在于工作站通常得要应付比较重要的公事应用， 例如工程界流体力学的数值模式运算、娱乐事业的特效功能处理、软件开发者的工作平台等等。 Linux 上面有强大的运算能力，以及支持度相当广泛的 GCC 编译软件， 因此在工作站当中也是相当良好的一个操作系统选择。

例如鸟哥所在的研究室目前就要将 Sun Unix 上面执行的大型模式移转到 Linux 上面， 据美国环保署内部人员的测试，发现 Linux 不但比较便宜 (X86 系统嘛！)而且速度还比较快呢！

- 桌上型计算机：

所谓的桌上型计算机，其实就是你我在办公室使用的计算机啦。一般我们称之为 Desktop 的系统。 那么这个 Desktop 的系统平时都在做什么呢？大概都是这些工作吧：

- 上网浏览；
- 文书处理；
- 网络接口之公文处理系统；
- 办公室软件(Office Software)处理数据；
- 收发电子邮件；

这些工作要被进行他需要什么东西在 Desktop 的计算机上面呢很简单， 『就是需要窗口』！ 因为上网浏览、文书编排的所见即所得接口，以及电子公文系统等等， 如果没有窗口接口的辅助，那么将对使用者造成很大的困扰。而众所皆知的， Linux 早期都是由工程师所发展的，对于窗口接口并没有很需要，所以造成 Linux 不太亲和的印象。

好在，为了要强化桌上型计算机的使用率， Linux 与 X Window System 结合了！ 如同前一章里面的说明，要注意的是， X Window System 仅只是 Linux 上面的一套软件， 而不是核心喔！所以即使 X Window 挂了，对 Linux 也可能不会有直接的影响呢！ 好，我们就来谈一谈 X Window System 吧！

- 由前一章提到的 Unix 与 Linux 的历史中，我们知道在 1986 年美美的窗口画面就已经在 Unix 上面出现过了，那个时候窗口画面被简称为 X 系统，而后来到了 1994 年的时候正式被整合在 Linux 里头！至于微软的 Windows 则是在 1995 年才出现！

- 所谓的 X Window System 就是以 XFree86 这个计划释出的 X11 这个窗口软件为管理显示核心的一套窗口接口的软件，我们常常简称他为 图形使用者接口(Graphical User Interface)。 这个 XFree86 只是 Linux 核心上面的一套软件而已， 他主要的工作就是管理图形接口输出的时候，几乎所有显示相关的硬件的控制， 例如显示卡、屏幕、键盘、鼠标、等等，都是 XFree86 管理的！或者， 我们可以称 XFree86 为 X-Window System 的服务器，简称为 X Server 。

- 至于我们所看到的美美的窗口画面，则是使用 X Server 提供的显示相关硬件的功能， 来达到图形显示的『窗口管理员(Window Mananger, WM)』所发挥的能力啦！ 这也就是说， WM 是挂在 X Server 上面来运作的一套显示窗口接口的软件，例如我们常见的 KDE, GNOME 等等都是 WM。

由于 Linux 整合了 X Window System ，虽然还有改善的空间，不过，却也已经具有相当个规模了！ 目前的 Linux 不但有强大美观的 KDE，以及 KDE 附加的 KOffice 办公室软件， 还有由 Sun 开发释出的自由软件 Start Office 以及修订过后的 Open Office 等办公室软件， 这些办公室软件同时也都拥有 文书处理、电子电子表格、简报软件等等，哇！功能太齐全了！ 也就是说，目前的 Linux 桌面应用上，已经可以应付大部分上班族群的工作需求了！

- 嵌入式系统：

近年来电子相关产业相当的蓬勃发展，其中，小型微电脑的发展甚为重要！ 例如家电产品、PDA、手机、数字相机以及其它微型的计算机配备。 这些计算机配备也都是需要操作系统来控制的！而操作系统是直接嵌入于产品当中的， 例如 PDA 本身就是一个小型的计算机操作系统啦！这些系统我们就称为嵌入式系统。

Linux 在这些嵌入式系统当中的应用是相当好的！因为 Linux 的核心的可变动性， 以及核心的小而美、效能佳的特性，让他在嵌入式设备的市场当中，具有很大的竞争优势！ Linux 的核心有多小呢？在您的 PC 上的 Linux （假设您已经安装了 Linux 了）核心最大绝对不会超过 2MB ，呵呵！够小了吧？而这个核心

里面还包含了很多可能用不到的模块，所以将所有不需要的功能移除,仅留下需要的程序,那么几百 KBytes 甚至几十 KBytes 的 Linux 核心 都可以被制作出来喔!所以啊,这对于嵌入式设备锱铢必较的内存空间来说， 真是相当的优秀啊!

网络服务器、工作站计算机、桌上型计算机等等，就是 Linux 目前最常被应用的环境了。 而您如果想要针对桌上型计算机，或者是网络服务器主机来学习的话，对于 Linux ， 您应该如何进行学习的课题呢?底下我们就来谈一谈。

基础学习

我们在 第零章 提到过 学习心态的分别 ，如果您看过的话,应该就不难理解,如果您仅想要了解 Linux ，并且利用 Linux 来作为您的桌上型计算机的话，那么，您只需要购买一本介绍 Linux 桌面设定，例如 中文输入法、打印机设定、因特网设定等等概念的的书籍即可， 不需要特别针对 Linux 来进行什么特殊的学习的！反正利用 Linux 的 X Window System 的图形接口就可以达到您的需求了！您可以选择专为桌上型计算机发行的 Linux distributions 例如：

- Ubuntu: http://www.ubuntulinux.org/
- Novell SuSE: http://www.novell.com/linux/suse/
- Mandriva: http://www.mandriva.com

但是仍须注意的是，SuSE 与 Mandriva 都有出多种版本，请挑选属于 Desktop 的那种喔！ 另外，您还可以参考一些网站的数据：

- 杨老师的图解桌面 http://apt.nc.hcc.edu.tw/docs/FC3_X/
- 中文指南 http://tw.ubuntuguide.org/

不过，如果您不想只学习 Linux 的桌上应用，还想学习更多 Linux 在网络上的应用， 那么单纯的以 X Window 来管理您的 Linux 主机，肯定是不够的！ 因为毕竟 X Window 是 Linux 上的一套软件，想用他来完全掌控 Linux 真的是很不容易的事情～ 而且，在服务器的应用上，档案的安全性、人员账号的管理、软件的安装/修改/设定、 登录文件的分析以及自动化工作排程与程序的撰写等等，都是需要学习的， 而且这些东西都还未涉及服务器软件呢！对吧！这些东西真的很重要，所以，您就得要这样学习才行：

从头学习

其实，不论学什么系统，『从头学起』是很重要的！还记得你刚刚接触微软的 Windows 都在干什么？还不就是由档案总管学起，然后慢慢的玩到控制台、玩到桌面管理， 然后还去学办公室软件，我想，你总该不会直接就跳过这一段学习的历程吧！？那么 Linux 的学习其实也差不多，就是要从头慢慢的学起啦！不能够还不会走路之前就想要学飞了吧！ ^_^!

常常有些朋友会写信来问鸟哥一些问题，不过，信件中大多数的问题都是很基础的！ 例如：『为什么我的使用者个人网页显示我没有权限进入？』、『为什么我下达一个指令的时候,系统告诉我找不到该指令？』、『我要如何限制使用者的权限』等等的问题,这些问题其实都不是很难的,只要了解了 Linux 的基础之后,应该就可以很轻易的解决掉这方面的问题呢！ 所以请耐心的，慢慢的，将后面的所有章节内容都看完。自然你就知道如何解决了！

此外，网络基础与安全也很重要，例如 IP 的基础，网络的 Gateway 设定基础与网络的相关概念！很多的朋友一开始问的问题就是『为什么我的 mail server 无法收到信件？』这种问题相当的困扰，因为发生的原因太多了，而朋友们常常一接触 Linux 就是希望『架站！』根本没有想到要先了解一下 Linux 的基础！这是相当伤脑筋的！尤其近来计算机怪客 （ Cracker ） 相当多， （ 真奇怪，闲闲没事干的朋友还真是不少.... ），一个不小心您的主机就被当成怪客跳板了！ 甚至发生被警告的事件也层出不穷！这些都是没能好好的注意一下网络基础的原因呀！

所以， 鸟哥希望大家能够更了解 Linux ，好让他可以为你做更多的事情喔！ 而且这些基础知识是学习更深入的技巧的必备条件呀！因此建议：

1. 先理解一下基础的硬件知识，不用一定要全懂(没那么多时间)，但是至少要『听过、有概念』即可；
2. 先了解一下 Linux 的基础知识，这些包含了『使用者、群组的概念』、『权限的观念』，『程序的定义』等等；
3. 必需至少学会一种以上的文书编辑器，例如最好学会通用版本的 vi 啰！
4. 实际操作 Linux 时，必定要学习的 Shell ，最好 Shell scripts 也能够了解；
5. 如果上面你都通过了，那么网络的基础就是下一阶段要接触的咚咚，这部份包含了『IP 概念』『路由概念』『TCP/IP』等等；
6. 如果连网络基础都通过了，那么网站的架设对你来说，简直就是『太简单啦！』

在一些基础知识上，可能的话，当然得去书店找书来读啊！ 如果您想要由网络上面阅读的话，那么这里推荐一下由 Netman 大哥主笔的 Study-Area 里面的基础文章，相当的实用！

- 计算机基础 (http://www.study-area.org/compu/compu.htm)
- 网络基础 (http://www.study-area.org/network/network.htm)

选择一本易读的工具书

一本好的工具书是需要的，不论是未来作为查询之用，还是在正确的学习方法上。 可惜的是，目前坊间的书大多强调速成的 Linux 教育，或者是强调 Linux 的网络功能， 却欠缺了大部分的 Linux 基础管理～鸟哥在这里还是要再次的强调，Linux 的学习历程并不容易， 他需要比较长的时间来适应、学习与熟悉，但是只要能够学会这些简单的技巧， 这些技巧却可以帮助您在各个不同的 OS 之间遨游！

您既然看到这里了，应该是已经取得了鸟哥的 Linux 私房菜 -- 基础学习篇 了吧！ ^_^ 。 希望这本书可以帮助您缩短基础学习的历程，也希望能够带给您一个有效的学习观念！ 而在这本书看完之后，或许还可以参考一下 Netman 推荐的相关网络书籍：

- 请推荐有关网络的书
  http://linux.vbird.org/linux_basic/0120howtolinux/0120howtolinux_1.php

不过，要强调的是，每个人的阅读习惯都不太一样，所以，除了大家推荐的书籍之外， 您必须要亲眼看过该本书籍，确定您可以吸收的了书上的内容，再下去购买喔！

实作再实作

要增加自己的体力，就是只有运动；要增加自己的知识，就只有读书；当然，要增加自己对于 Linux 的认识，大概就只有实作经验了！所以，赶快找一部计算机，赶快安装一个 Linux 套件，然后快点进入 Linux 的世界里面晃一晃！相信对于你自己的 Linux 能力必然大有斩获！除了自己的实作经验之外，也可以参考网络上一些善心人士整理的实作经验分享喔！ 例如最有名的 Study-Area（http://www.study-area.org）等网站。

此外，人脑不像计算机的硬盘一样，除非硬盘坏掉了或者是数据被你抹掉了， 否则储存的数据将永远而且立刻的记忆在硬盘中！在人类记忆的曲线中， 你必须要『不断的重复练习』才会将一件事情记得比较熟！同样的，学习 Linux 也一样，如果你无法经常摸索的话，那么，抱歉的是，学了后面的， 前面的忘光光！学了等于没学，这也是为什么鸟哥当初要写『鸟哥的私房菜』这个网站的主要原因， 因为，我的忘性似乎比一般人还要好～～呵呵！所以，除了要实作之外，还得要常摸！才会熟悉 Linux 而且不会怕他呢！

好了，底下列出几个学习网站来提供大家做为参考实作的依据： （注：由于不同的网站当初撰写的时候所用的 Linux 套件或版本与目前的主流并不相同，因此参考他人的实作经验时， 必须要特别留意对方的版本，否则反而可能造成你的困扰喔！）

- Study-Area http://www.study-area.org
- 鸟哥的私房菜馆 http://linux.vbird.org
- 狼主的网络实验室 http://netlab.kh.edu.tw/index.htm
- 卧龙大师的网络技术文件 http://linux.tnc.edu.tw/techdoc/
- 大南国小（林克敏主任文件集） http://freebsd.lab.mlc.edu.tw/
- 张毓麟先生的小文章 http://www.se.ntou.edu.tw/~ylchang/MyDocuments/index.html
- 台湾 Linux 社群 http://www.linux.org.tw/
- 吴仁智的文件集 http://www.cses.tcc.edu.tw/~chihwu/

学习的方法

如果您想透过自学来学习 Linux 的话，那么努力的实作之外， 还需要学的有效的方法。首先，我们就刚刚也稍微提到的 X Window 与 command line 的议题来继续讨论。

X Window 还是 command line

由前面的介绍我们可以知道，虽然目前 X-Window 的接口越做越漂亮， 而且也已经渐渐的可以来控管整个系统了！但是必须要注意的是， X-Window 毕竟还只是一个 Linux 上面的软件，并不是一套『操作系统』，所以实际上使用他来设定系统的时候，还是有相当多的困扰的，因为毕竟他无法完全的管理好我们的 Linux 啊！

虽然就以 Desktop 的型态来说，X-Window 是让 Linux 立刻深入人心的方法。不过， X-Window 在使用的时候还是有相当多的问题的，最大的问题来自于 『系统资源的有效应用』，以 鸟哥的使用情况来说，我的系统资源并没有很好，但是 X-Window 本身相当的消耗系统资源，如果一开 X-Window ，那么你的内存几乎都被 X-Window 吃光了！您要如何来用剩下的系统资源来进行高效率的其它工作呢？！ 这也是为什么

很多的书籍与网站都会希望使用者架设网站的时候，不要启动 图形使用者接口的原因啰！

以下再来说说 X-Window 学习与 command line 学习的角度。

- X-Window

如果您对于 Linux 的要求是『桌上型计算机』，并且你又不架设网站的话，那么学习 X-Window 对您而言，绝对是需要的！至于指令列模式对你就不是这么必要了！但是，如果 Linux 对你而言是『服务器与工作站』的话，那么 X-Window 可能就不是这么重要， 但是指令列模式可就大大的重要啦！

因为，如果以 X-Window 作为学习 Linux 的方式，那么未来一定会有死角， 这是因为 X-Window 了不起也只是 Linux 的『一套软件』而不是『Linux 核心』 此外，目前发展出来的 X-Window 对于系统的管理上还是有无法掌握的地方， 举个例子来说，如果 Linux 本身捉不到网络卡的时候， 请问如何以 X-Window 来捉这个硬件呢？!还有，如果需要以 tarball 的方式来安装软件并加以设定的时候，请以 X-Window 来架设他！这可能吗？当然可能，但是这是在考验『X-Window 开发商』的技术能力， 对于了解 Linux 架构与核心并没有多大的帮助的!所以说,如果只是想要『会使用 Linux 』的角度来看，那么确实使用 X-Window 也就足够了，反正搞不定的话， 花钱请专家来搞定即可；但是如果想要更深入 Linux 的话，那么指令列模式才是不二的学习方式！

- 服务器端

如果 Linux 对你而言是『生财』的工具，呵呵！那可不是只要学习 X-Window 能够解决的了！举个例子来说好了，假如你的客户人在台北，而你人在远方的台南， 他的 Linux 服务器出了问题，要你马上解决他，请问：要您亲自上台北去修理？ 还是他搬机器下来让你修理？！或者是直接请他开个账号给你进去设定即可？！ 想当然尔，就会选择开账号给你进入设定即可啰！因为这是最简单而且迅速的方法！然而这个方法使用的方式却不是 X-Window 作的到的！因为 X-Window 太耗资源，实在不容易让您这样子联机（很麻烦的啦！） 所以啰！文字界面是相当重要的！尤其如果想要深入 Linux 的核心时，那么以文字界面来了解 Linux 就更需要了！所以说，不要怕麻烦，还是多摸一些文字界面的东西吧！！ 帮助会比较大哩！

所以基本上， VBird 还是希望大家可以多多的以 文字接口(command line)的方式来学习 Linux 啦！

主机/网络数据查询

其实，在 Linux 主机及网络上面已经有相当多的 FAQ 整理出来了！ 所以，当你发生任何问题的时候，除了自己检查， 或者到上述的实作网站上面查询一下是否有设定错误的问题之外，最重要的当然就是到各大 FAQ 的网站上查询啰！以下列出一些有用的 FAQ 网站给您参考一下：

- Linux 自己的文件数据： /usr/share/doc (在你的 PC 中)
- CLDP 中文文件计划 http://www.linux.org.tw/CLDP/
- Unix 的一般常见问题 http://www.csie.nctu.edu.tw/document/unixfaq/
- The Linux Documentation Project：http://www.tldp.org/

上面比较有趣的是那个 LDP (Linux Documentation Project)， 他几乎列出了所有 Linux 上面可以看到的文献数据，各种 How-To 的作法等等，虽然是英文的， 不过，很有参考价值！

除了这些基本的 FAQ 之外，其实，还有更重要的问题查询方法，那就是利用酷狗 (Google) 帮您去搜寻答案呢！在鸟哥学习 Linux 的过程中，如果有什么奇怪的问题发生时， 第一个想到的，就是去

http://www.google.com.tw 搜寻是否有相关的议题。 举例来说，我想要找出 Linux 底下的 NAT ，只要在上述的网站内，输入 Linux 跟 NAT， 立刻就有一堆文献跑出来了！真的相当的优秀好用喔！ 您也可以透过酷狗来找鸟哥网站上的资料呢！ http://linux.vbird.org/Searching.php

---

真的没办法，发问吧！

如果自己真的都找过了相关的信息，却还是无法得到答案时，只好去网络上面求救了。 但是，不要直接问问题呢～发问之前，务必注意过底下这些事情：

- 注意讯息，自行解决：

一般而言， Linux 在下达指令的过程当中，或者是 log file 里头就可以自己查得错误信息了，举个例子来说，当您下达：

```
ls -l /vbird
```

时，由于系统并没有 /vbird 这个目录，所以会在屏幕前面显示：

```
ls: /vbird: No such file or directory
```

这个错误讯息够明确了吧！系统很完整的告诉您『查无该数据』！ 呵呵！所以啰，请注意，发生错误的时候，请先自行以屏幕前面的信息来进行 debug (除错)的动作，然后，如果是网络服务的问题时，请到 /var/log 这个目录里头去查阅一下 log file (登录档)，这样可以几乎解决大部分的问题了！

- 搜寻问题的信息：

一般来说，如果发生错误现象，一定会有一些讯息对吧！ 那么当您要请教别人之前，就得要将这些讯息整理整理， 否则网络上人家也无法告诉您解决的方法啊！这一点很重要的喔！

- 讨论区的提问：

万一真的经过了自己的查询，却找不到相关的信息，那么就发问吧！ 不过，在发问之前建议您最好先看一下『 提问的智慧 http://phorum.vbird.org/viewtopic.php?t=96』 这一篇讨论！然后，你可以到底下几个讨论区发问看看：

- Study-Area 讨论区 http://phorum.study-area.org
- 鸟哥的私房菜馆讨论区 http://phorum.vbird.org
- 狼主的网络实验室讨论区 http://netlab.kh.edu.tw/board/board.asp
- telnet://bbs.sayya.org

不过，基本上去每一个讨论区回答问题的熟手，其实都差不多是那几个， 所以，您的问题『不要重复发表在各个主要的讨论区！』 举例来说，鸟园与酷学园讨论区上的朋友重复性很高，如果您两边都发问， 可能会得到反效果，因为大家都觉得，另外一边已经回答您的问题了呢～～

此外，发问的时候一定要注意到某些礼节！最好是先以搜寻的方式搜寻一下 该讨论区是否有您需要的文章之后，在发问！这样可以获得事半功倍的功能喔！此外， Netman 兄提供的一些学习的基本方针，提供给大家参考：

- 在 Windows 里面，程序有问题时，如果可能的话先将所有其它程序保存并结束， 然后尝试按救命三键(Ctrl+Alt+Delete)，将有问题的程序(不要选错了程序哦) “结束工作”，看看能不能恢复系统。不要动不动就直接关机或 reset。

- 有系统地设计档案目录，不要随便到处保存档案以至以后不知道放哪里了， 或找到档案也不知道为何物。
- 养成一个做记录的习惯。尤其是发现问题的时候，把错误信息和引发状况以及 解决方法记录清楚，同时最后归类几定期整理。别以为您还年轻， 等你再弄多几年计算机了，您将会非常庆幸您有此一习惯。
- 如果看在网络上看到任何好文章，可以为自己留一份 copy，同时定好题目，归类存档。
- 作为一个使用者，人要迁就机器；做为一个开发者，要机器迁就人。
- 学写 script 的确没设定 server 那么好玩，不过以我自己的感觉是： 关键是会得“偷”，偷了会得改，改了会得变，变则通矣。
- 在 Windows 里面，设定不好设备，您可以骂它；在 Linux 里面，如果设定好设备了，您得要感激它！

鸟哥的建议：

除了上面的学习建议之外，还有其它的建议吗？确实是有的！其实， 无论作什么事情，对人类而言，两个重要的因素是造成我们学习的原动力：

- 成就感
- 兴趣

很多人问过我，鸟哥是怎么学习 Linux 的？由鸟哥之前的 自我介绍 与对于 Linux 的接触历程，你大概会知道， 原来我本人对于计算机就蛮有兴趣的，加上工作的需要，而鸟哥又从中得到了相当多的成就感，所以啰，就一发不可收拾的爱上 Linux 啰！因此，鸟哥个人认为，学习 Linux 如果玩不出兴趣，他对你也不是什么重要的生财工具，那么就不要再玩下去了！ 因为很累人ㄋㄟ～而如果你真的想要玩这么一套优良的操作系统， 除了前面提到的一些建议之外，说真的，得要培养出兴趣与成就感才行！ 那么如何培养出兴趣与成就感呢？可能有几个方向可以提供给你参考：

- 建立兴趣：Linux 上面可以玩的东西真的太多了， 你可以选择一个有趣的课题来深入的玩一玩！不论是 Shell 还是图形接口等等，只要能够玩出兴趣，那么再怎么苦你都会不觉得喔！
- 成就感：成就感是怎么来的？说实在话， 就是『被认同』来的！怎么被认同呢？写心得分享啊！当你写了心得分享，并且公告在 BBS 上面，自然有朋友会到你的网页去瞧一瞧，当大家觉得你的网页内容很棒的时候， 哈哈！你肯定会加油继续的分享下去而无法自拔的！那就是我啦...... ^_^！而且，就鸟哥的经验来说，你『学会一样东西』与 『要教人家会一样东西』思考的纹路是不太一样的！ 学会一样东西可能学一学会了就算了！但是要『教会』别人，那可就不是闹着玩的！得要思考相当多的理论性与实务性方面的咚咚，这个时候，你所能学到的东西就更深入了！ 鸟哥常常说，我这个网站对我在 Linux 的了解上面真的的帮助很大！
- 协助回答问题： 另一个创造成就感与满足感的方法就是『助人为快乐之本！』当你在 BBS 上面告诉一些新手，回答他们的问题，你可以获得的可能只是一句『谢谢！感恩呐！』 但是那句话真的会让人很有快乐的气氛！很多的老手都是因为有这样的满足感， 才会不断的协助新来的朋友的呢！此外，回答别人问题的时候，就如同上面的说明一般， 你会更深入的去了解每个项目，哈哈！又多学会了好多东西呢！
- 参与讨论： 参与大家的技术讨论一直是一件提升自己能力的快速道路！因为有这些技术讨论， 你提出了意见，不论讨论的结果你的意见是对是错，对你而言，都是一次次的知识成长！这很重要喔！ ^_^。目前活动力很高的台南 Linux 使用者社群 (Tainan Linux User Group, TnLUG) 在每

个月都会举办研讨会，目前每个月在台南与台北两地均有举办呢！有兴趣参加讨论的朋友可以看看： http://tnlug.linux.org.tw/

此外，除了这些鸟哥的经验之外，还有在 BBS 上面有一封对于 Linux 新手相当有帮助的文件数据，大家可以多看一看：

- 李果正先生之 GNU/Linux 初学者之旅： http://info.sayya.org/~edt1023/linux_entry.html 鸟哥这里有也一个备份 http://linux.vbird.org/linux_basic/0120howtolinux/0120howtolinux_3.php
- 信息人的有效学习(洪朝贵教授网页) http://www.cyut.edu.tw/~ckhung/published/018learn.shtml

除了这些基本的初学者建议外，其实，对于未来的学习，这里建议大家要『眼光看远！』！！ 一般来说，公司行号会发生问题时，他们绝不会只要求各位『单独解决一部主机的问题』而已， 他们需要的是整体环境的总体解决『Total Solution』。 而我们目前学习的 Linux 其实仅是在一部主机上面进行各项设定而已， 还没有到达解决整体公司所有问题的状态。当然啦，得要先学会 Linux 相关技巧后， 才有办法将这些技巧用之于其它的 solution 上面！

所以，大家在学习 Linux 的时候，千万不要有『门户之见』，认为 MS 的东西就比较不好～ 否则，未来在职场上，竞争力会比人家弱的！有办法的话， 多接触，不排斥任何学习的机会！都会带给自己很多的成长！ 而且要谨记：『不同的环境下，解决问题的方法有很多种，只要行的通，就是好方法！』

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 我的计算机上面老是出现问题，他会有一个错误讯息为 『fatal: SASL per-connection security setup』 请帮我找出可能的原因为何？

  先跑到 http://www.google.com.tw 里面去，输入上列的错误讯息， 就可以找到很多文件，根据文件去判断吧！

- Windows 的操作系统当中，老是自动出现一个名为 internet optimizer 的软件， 我想要知道他是什么，可以怎么找？

  利用 http://www.google.com.tw 输入 inetnet optimizer 后， 就可以找到相关的信息。基本上，这是一个木马程序啦！赶紧移除吧！

- 我的 Linux 发生问题，我老是找不到正确的答案，想要去 http://phorum.study-area.org 提问，应该要先做哪些动作才发问？

  1. 先将您 Linux 上面的问题作一个清楚的描述，例如，做了什么动作，结果发生了什么讯息与结果。
  2. 先到 http://phorum.study-area.org 内的『搜寻』查询有无相关的问题
  3. 再到 http://www.google.com.tw 查询一下有无相关的信息

4. 将您的问题描述写下，并且写下您的判断，以及查询过数据的结果。
5. 等待回复～

- 你觉得学习 Linux 最重要的一环是什么？

  其实是自己的学习心态～最重要的地方在于能够『刻苦耐劳～』 ^_^

- 什么是 LDP ？全名为何？网站在哪里？

  LDP 是 Linux Documentation Project 的缩写，内容提到的是 Linux 操作系统的各个 How-To 以及相关的说明文件如 man page 等等。网站在 http://www.tldp.org 喔！

- 想一想再回答，为何您想要学习 Linux ？有没有持续学习的动力？？ 您想要 Linux 帮您达成什么样的工作目标？

Linux 主机的规划

最近更新日期：2005/06/15

一部好的 Linux 主机系统，除了后续的维护之外，一开始的硬件选择与 distributions 的搭配， 以及主机预期的『工作任务』来加以思考，而选择最合适的硬件，这是很重要的一个开始！ 俗话说『钱要花在刀口上』，没有必要为了一个小小的 IP 分享的功能来买一部双 CPU 的硬件架构吧？ 而一部简单的个人计算机，也真的无法满足中大型企业的工作环境需求。 在这一章里面，鸟哥会向您介绍一下，在开始安装 Linux 之前，您应该要先思考哪些工作？ 好让您后续的主机维护轻松愉快啊！ 此外，要了解这个章节的重要性，您至少需要了解到 Linux 档案系统的基本概念， 所以，在您完成了后面的相关章节之后，记得要再回来这里看看如何规划主机喔！ ^_^

认识主机各部分硬件组件

在开始安装 Linux 之前，有些功课是要先作的！因为 Linux 对于较新的硬件的支持度可能比较不足， 所以，您必须要了解您的主机是否为 Linux 所支持的 CPU、RAM、显示卡、网络卡等等。 此外，您也必须要先了解到您的 Linux 预计想要达成的功能是什么？这样在选购硬件时， 才会知道，那个部分是最重要的啊！举例来说，桌上型的使用者，应该会用到 X Window 系统， 此时，显示卡的优劣与内存的大小可就占有很重大的影响。如果是想要做成档案服务器， 那么硬盘或者是其它的储存设备，应该就是您最想要增购的组件啰！所以说，功课还是需要作的啊！

鸟哥在这里要不厌其烦的再次的强调，Linux 对于计算机各组件/装置的分辨，与我们惯用的 Windows 系统完全不一样！因为，各个组件或装置在 Linux 底下都是『一个档案！』 这个观念我们在 Linux 是什么的章节里面已经提过， 这里我们再次的强调。因此，您在认识各项装置之后，学习 Linux 的装置代号之前， 务必要先将 Windows 对于装置名称的概念先拿掉～否则会很难理解喔！

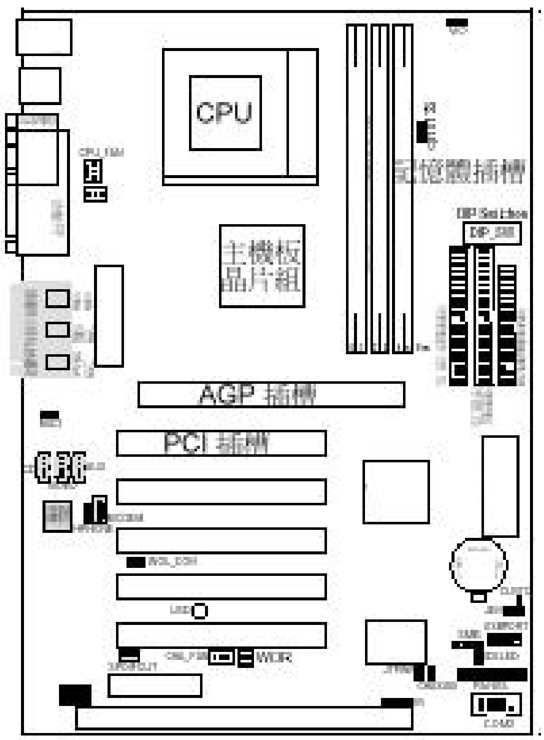
认识计算机的硬件配备

『什么？学 Linux 还得要玩硬件？！』呵呵！没错！ 因为 Linux 早期是与 x86 架构的个人计算机系统紧密结合，而且我们由前两章的 Linux 是什么 当中也晓得，硬件与操作系统的关系可是很大的！ 所以，

我们当然就得要了解一下个人计算机主机的基本架构了！

事实上，计算机主机的硬件配备与这部主机未来的功能是很有相关性的！ 举例来说，家里有小孩，或者自己仍然算是小孩的朋友大概都知道：『 要用来打 Game 的 "游戏机计算机" 所需要的配备一定比办公室用的 "工作机计算机" 配备更高档』，为什么呢？因为『计算机游戏』所需要的显示、 CPU 等资源较高，所以需要适合比较大量运算的配备，因此等级上面不能太差啊(例如 Athlon 或者是 P4 系列) 至于办公室个人计算机环境中，通常我们不太可能会使用到太复杂的软件，所以，等级不需要太好 (例如K6-2 500)，就可以运作的很顺畅啦！那什么叫做高档配备？ 呵呵！底下我们得先来约略了解一下计算机主机里面的各个配备， 以及他们对于系统性能的影响啰！

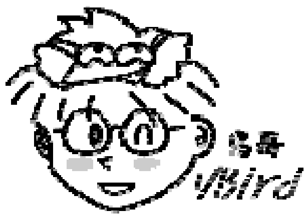
所谓的『计算机主机』就是那个机壳内的咚咚～当您拆掉主机机壳之后，会发现里面有一张大板子， 这块大板子上面插了好几块小板子，还有一颗小风扇安置在一块四方形的东西上面。 此外，还有几颗硬盘与光盘或软盘被锁在机壳里面，还有一个铁盒子，上面有很多电线跑出来， 这些电线的接头都被接在硬盘/光盘/软盘以及大板子上面。其中最重要的就是那张大板子， 因为所有的机壳内的东西，都会接到这个大板子上面！这个大板子就是『主机板』。 主机板有点像底下这个样子：



图一、主机的架构示意图

上图一中，中间上面的部分就是中央处理器 （ CPU ） 的插槽了，至于 CPU 右边那三个直直的则是内存 （ RAM ） 的插槽； 而在中间偏下有几个名称为 PCI 的咚咚，那就是一些适配卡的插槽了。包括 SCSI 适配卡、网络卡，还有一些声卡等等，都是接在 PCI 卡上面的！至于在 PCI 上面还有一个 AGP 插槽是干嘛的？ 那个就是前一阵子的主流显示卡接口噜！此外， 在内存的右边还有几支黑黑的咚咚，那就是使用『排线』连接主机板与硬盘的接口插槽， 我们称他为『IDE 接口』， 而硬盘除了 IDE 接口外，现在比较流行的应该是 SATA 接口，这个接口的硬盘插槽比较小一点， 连接线也比较好接，因为比较细，对于机壳内的空气对流比较好。 这个部分相当的重要，等一下我们会再提到的！底下我们来谈一谈关于几个主要的组件，然后再介绍整体的运作吧！

Tips:
很讨厌的是，个人计算机的发展不断的向上延伸，各项接口也在不断的改善，截至目前为止 (2005/06)，有 PCI Express 、AGP 渐渐被淘汰、IDE 接口被 SATA 接口所取代，内存也均已 DDR 接口取代原本的 PC133 的 SDRAM，这些接口的改善也造成您的硬件在升级时候的困扰，因为，很多旧的配备无法被重复利用的！ 所以，上面的图示仅是一个简略的介绍！您需要随时请教店家喔！



- 中央处理器 (CPU) ：

CPU 可以说是一部计算机主机里面相当重要的东西了，因为， 他负责了所有事件的运算！而且，跟大量运算有关的主机，主要的关键几乎就是卡在 CPU 的速度上。 目前全世界前两大的个人计算机 CPU 制造商为 Intel 与 AMD ，而这两家公司自己的 CPU 规格就不少了，加上有两家制造商，哇！这表示， CPU 的规格多的吓人～

早期的 CPU 规格都是由 Intel 来拟定，然后大家按照他的规格去设计自己的 CPU ， 所以，在主机的购置上面，会比较不容易产生不兼容的情况。 但因为某些因素，导致 AMD 自行在 x86 的架构上发展自己的 CPU 脚位， 而且因为 CPU 结构的变更，使得脚位的定义越来越多。 目前因为规格太多， 而且 CPU 的插脚的脚位都不一样，有的即使一样但是 CPU 的运作电压不同， 也无法兼容！因此，在选购 CPU 与相关的主机板时， 务必要询问正确的信息， 否则买了 CPU 与主机板不能兼容的话，那么两个东西都会变成废铁的喔！

目前流行的 CPU 规格大致分为 Intel 的 P-4 系列，而 P-4 系列未来又分为双核心与单核心， 这两款 CPU 是不兼容的，主机板也不兼容，必须特别留意。至于 AMD 的 Sempron, Athlon64, Athlon64x2 当中，Sempron 是入门级， Athlon64 与 Athlon64x2 是 64 位的 CPU， 分别是单核心与双核心，两款的脚位相同，据说主机板是可以互通的！但是 Sempron 与 Athlon64 的主机板则不兼容！要特别注意。(注：所谓的双核心，是在一颗 CPU 里面， 建构了两个运算单元，也就是说，这个双核心的 CPU 有两个可以运算的实体 CPU 就是了。 而 AthlonXP 也已经被新的型号 Sempron 所取代了！)

好了，那么 CPU 的速度除了核心的优良与否之外，常用来判断同级产品之间的速度，就是频率了。 所谓的频率，简单的说，就是单位时间的运算次数。 所以，频率越高，代表这个装置设备的运算次数越多次，当然速度上就会比较快。 (注：不同的 CPU 之间不可以单纯用频率来判断运算的效能喔！例如 AMD 的 Athlon64 2G 在运作上，效能比 P-4 2G 还要好很多！所以，频率目前仅能用来比较同样的 CPU 的速度。) 比较特别的是， CPU 有所谓的倍频与外频，有什么意义呢？

- CPU 有所谓的『倍频』与『外频』， 外频是 CPU 与接口设备进行数据传输/运算的速度， 至于倍频则是 CPU 本身运算时候加上去的一个运算速度！ 两者相乘才是 CPU 的频率。与 CPU 外频有关的咚咚为内存与主机板芯片组。 一般来说，越快的频率代表越快的 CPU 运算速度。以 Intel 的 PIII 频率 933 MHz 为例，
    - CPU 外频与倍频： 133(外频) X 7(倍频) MHz
    - RAM 频率：通常与 CPU 之外频相同，为 133 MHz
    - PCI 接口(包含网络卡、声卡等等的接口喔) 133/4=33 MHz
    - AGP 界面 ： 133 / 2 = 66 MHz (这是 AGP 正常的频率喔！)

- 外频是可以超频的！ 什么是超频呢？原本的 CPU 外部频率假设是 133 好了，如果您藉由某些工具， 或者主机板本身也可能会提供这个工具，那您就可以将 133 提升到比较高的频率， 那就是所谓的超频。为什么要超频呢？因为可以在比较便宜的 CPU 上面让频率升到比较高， 等于是『赚到了』的意思。不过，超频本身的风险很高～ 如果是超外频的话，例如到 166 MHz 时，你的 AGP 将达 (166/2=83) 而 PCI 也将达 (166/4=41.5) ，高出正常值甚多，通常， 越快的外频由于所有的设备运作频率都会提升，所以，可以让效能提高不少， 但也可能会造成系统不稳定！例如常常当机，或者是造成某部分组件的寿命简短等等。 此外，目前的计算机系统真的是够快了，不需要超频啦！而且，在 Linux 系统中，『不建议超频』，因为，即使 CPU 可以耐的住这么高的频率，但是系统的运作是全面性的，只要有一个设备当机， 那么你的系统就跟着当掉啦！而因为超频之后，系统频率高出正常值太多， 所以当然容易造成不稳定呢！
- 另一个需要注意的是，『CPU 是有分等级的』，而目前很多的程序都有对 『CPU 做最佳化』的行动，所以就会有所谓的 i386, i586, i686 为附档名的档案产生啦！基本上，在 P MMX 以及 K6-III 都称为 586 的 CPU ，而 Intel 的赛扬以上等级与 AMD 的 K7 以上等级，就被称为 686 的机器了！万一改天你发现一些程序是注名给 686 的 CPU 使用时，就不要将他安装在 586 以下等级的计算机中，否则可是会无法执行该软件的！不过，在 686 倒是可以安装 386 的软件喔！也就是说，这些东西具有向下兼容的能力啦！

- 内存 (RAM) ：

内存对于系统来说，真是一个重要的家伙，怎么说呢？刚刚提到，计算机真正运作的核心是 CPU ，但是真正『喂给』 CPU 运算数据的，那就是内存 (Memory, RAM) 啦！ 所以你的操作系统的核心啦、软硬件的驱动程序啦、所有你要读取的档案啦等等的， 都需要先读入内存之后，才喂给 CPU 来进行数据的运作！您瞧！ RAM 可重要的很吧！

此外，一些比较优良的操作系统，也会将常用的档案或程序等数据， 给他常驻在内存内而不直接移除，如此一来，下次取用这个数据时， 就不需要在去周边存取设备读取一次，呵呵！对于系统速度来说，真是不无小补喔！ 所以啰，您就会晓得，如果你常常开启大容量的档案，以及执行一些很占资源的软件， 那么你就必须要『很大量的内存』来帮助你存放这些数据，瞧！很重要的一个项目吧！

内存目前的规格也不少，主要有两种，分别是 SDRAM 与 DDR，新一代的内存通常使用 DDR 这种规格的内存， 不过还得配合主机板与 CPU 来选择 RAM 的规格才行！对于一个系统来说， 通常越大的内存代表越快速的系统，这是因为系统不用常常释放一些内存内部的数据。 以服务器来说，内存的容量有时比 CPU 的速度还要来的重要的！

- 显示卡 (VGA card)：

显示卡对于图形接口有相当大的影响！因为我们要将影像数据显示到屏幕时， 就需要使用到显示卡 (VGA Card) 的相关硬件功能了。 目前 3D 的画面在计算机游戏接口与工作接口很大量的被使用，而由于如果这些 3D 画面没有先经过处理而直接进入 CPU 来做处理的话，将会影响到整体运作的速度，因为 CPU 的工作实在太多了！这个时候就有所谓的 GPU 出现了！

GPU 那是什么咚咚呢？为了避免由于大量的 3D 画面造成 CPU 的困扰，所以显示卡开发商就在显上卡上面安插一个可以处理这些很耗 CPU 运算时间的硬件来处理这些画面数据，如此一来，不但图形画面处理的速度增快了， CPU 的资源也会多出来以执行其它的工作喔！

目前的显示卡也有两种主要规格，一种是以传统 AGP 接口来进行影像数据的传输， 一种则是以更快的 PCI Express 接口来传输数据！由刚刚我们提到的 CPU 运作频率中，我们可以知道 PCI 的接口标准速度是

33MHz ，但是 AGP 标准是 66 MHz 。不过，即使是 AGP 的 66 MHz 也无法满足现在的需求了，因此， 才又有 PCI Express (简称 PCI-E) 接口出现。这个接口的速度又比 AGP 来的更加的快速呢。 不过，您到底要买哪一款？还是得要看您的主机板有没有支持该接口才行！

另外， VGA 卡上面也有一个内存，这个内存的大小可以影响您屏幕输出的分辨率与画素喔！ 这个内存是直接嵌入于显示卡上面的，与你的主存储器(上面提到的 RAM)没有关系！ 一般来说，服务器没有 X Window 的话，显示卡并不重要，如果是需要使用到图形接口的话， 那么这个显示卡内存的容量就比较重要了！

- 硬盘与存取装置(hard disk)：

总是需要有数据，我们的主机才能够藉由这些数据来加载，来运作吧？ 这些数据一般来说，就是存放在主机的硬盘上面了。而我们也可以透过可携式储存媒体， 例如光盘、Zip 磁盘、软盘片等等来传递数据的。 我们就单纯来说说硬盘好了。 在个人计算机上面，主流的硬盘存取接口应该是 SATA 与 IDE 这两种。

一般来说，主机板上面至少应该都会有两个 IDE 或者 SATA 的插槽， 而每个插槽都可以接两个 IDE 或者 SATA 接口的硬盘或装置。 SATA 是近年来开发出来的新接口，他的硬盘转速比较高，存取效能要比传统的 IDE 接口来的好。 此外， SATA 的特色就是，他与主机板连接的排线可以比较长(可长达 1m)， 并且排线比较细，可以帮助主机机壳内部的通风，有很不错的效果。 在 Linux 上面， SATA 或 IDE 接口的命名方法都是一样的， 所以未来我们还是以 IDE 来介绍装置。

由于一个 IDE 插槽可以接两个 IDE 接口的装置，那么系统怎么知道那个是那个？此时就需要 IDE 装置的跳针 (Jumper) 来设定了！你可以在一个 IDE 接口接的两个装置上面，以排线接一个 Master 以及一个 Slave 的装置！而 Master 与 Slave 可以在任何一个 IDE 装置上面找到的！也就是说，如果你有两颗硬盘，那么你可以将任何一颗调成 Master ，但是另外一颗则必须为 Slave 才行！否则 IDE 接口会无法分办，而造成系统的当机喔！至于硬盘的一些相关数据我们在后面的章节再来提！

至于硬盘的选购上面，您除了必须要注意硬盘的容量大小之外， 还得知道硬盘的转速，以及缓冲存储器的大小。目前的要求是，转速至少得 7200 转， 缓冲存储器最好可以选择 8M 比较好一些。

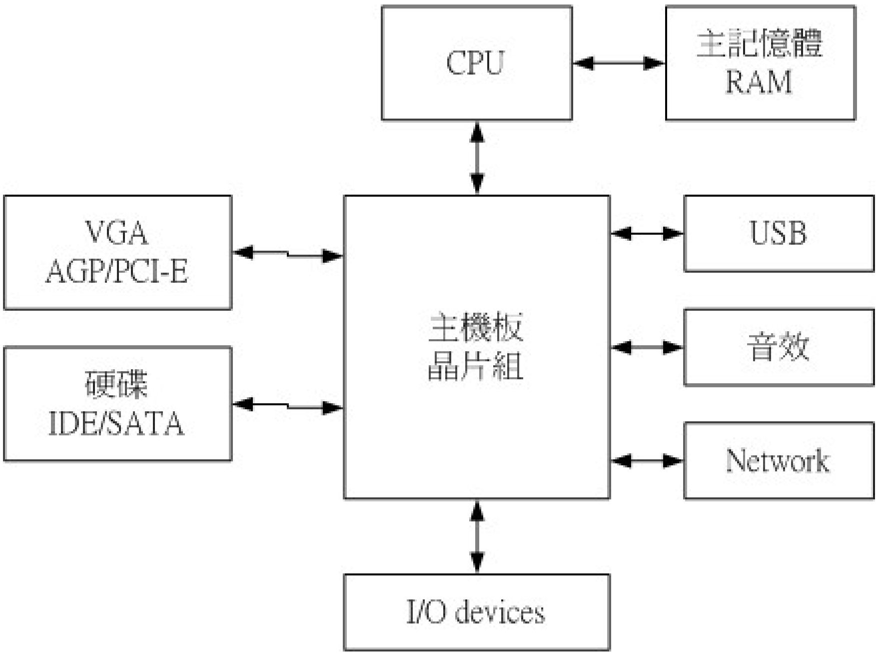
- PCI 适配卡：

我们常用的网络卡、声卡、特殊功能卡等等，几乎都是透过 PCI 插槽来安装的， 这些东西就被称为 PCI 接口的装置啦！当然不止，因为主机板上还有很多内建的 PCI 装置呢！

  - 网络卡：网络卡很重要吧！因为他是让你可以连接上 Internet 的重要功臣！网络卡的传输速度目前都已经可以支持 10/100Mbps 的主流速度了！但是网卡的好坏却差很多！同样是 10/100Mbps ，但是 Intel 与 3Com 的卡硬是要比一般的杂牌卡多出 1000 元新台币以上，原因无他，因为网络卡的稳定性、消耗 CPU 资源的特性与其它特殊功能等，Intel 与 3Com 要比其它的厂牌优良的多！ 至于网络线连接的接口上面，目前则几乎已经都是 RJ-45 的接口插孔了， 这种接口有点像是电话线插孔，不过要稍微大一点。 另外，由于网络的需求不断向上攀升，所以，网络卡已经有很多 Gigabits 的速度出现了。 您也可以选购 10/100/1000 Mbps 的网络卡喔！
  - SCSI 适配卡：这个东西可以用来连接 SCSI 的接口的装置！以硬盘为例，目前的硬盘除了个人计算机主流的 IDE/SATA 接口(刚刚说过了) 之外，就是这个 SCSI 接口！由于 SCSI 接口的装置比较稳定， 而且装置的运转速度较快，因而速度也会快的多，而且也比较不耗费 CPU 的资源。目前 SCSI 适配卡是一般大型服务器主机的硬盘传输接口，不过，用在个人计算机上面的话，IDE/SATA 接口就够了！因为， SCSI 接口的硬盘很贵呢！

- 主机板(Mainboard 或 Mother board)：

我们在 图一 提到的那个主机板真是重要！ 上面提到的 CPU、RAM、VGA Card、PCI Card等等，全部都是接在这个主机板上面的。 当然啦，这个主机板就得要负责沟通所有接口的工作了。而沟通所有上面提到的咚咚的东西， 就是 主机板的芯片组。 由于主机板上面的芯片组将负责与 CPU、RAM 及其它相关的输出、输入装置， 所以，芯片组设计的好坏也相差甚多喔！整个主机板芯片组与各个组件之间的沟通可以 使用下图二来简单的说明：



图二、芯片组架构示意图

先要提醒您的是，我们这里仅列出芯片组功能示意而已，并没有完整列出芯片组的详细架构喔。 底下我们就来提一下芯片组的相关功能介绍吧！

- 芯片组的功能：芯片组就是在沟通 CPU、 RAM 、输出与输入装置的重要角色！还记得刚刚我们在 CPU 时候提到的频率问题吧？在这里我们再次的强调，CPU 的外频就是芯片组与其它周边沟通的速度啰，假如使用刚刚的 P-III 933 MHz 做为例子的话，那么你的芯片组运作频率应该是以 133 为基准，所以 CPU 与芯片组沟通是 133 ，芯片组与 RAM 亦是 133 的速度，与 PCI 接口则通常是 33 (133/4)，而与 AGP 则是 66 (133/2)啰！所以，一个芯片组就需要负责这么多不同的频率操作喔！ 呵呵！所以啰，芯片组的好坏对于系统的影响也是相当大的！ 另外，目前很多的技术可以提升各个与芯片组之间沟通的频率速度，例如 DDR 内存，可以将频率再提升一倍，所以，如果刚刚是 133 ，那么使用 DDR 内存时，就可以提升成为 266 啰！至于 P4 则芯片组与 CPU 之间则可以提升四倍的频率速度，遗憾的是，芯片组能支持的规格， 只有一种，并无法支持所有的规格，也就是说，你的 Intel 芯片组的主机板，只能支持 Intel 的 CPU 与芯片组能沟通的内存规格；
- CPU、内存与芯片组 (就是主机板啦) 在选购的时候需要一起考虑，因为芯片组(主机板)能够支持的 CPU 只有特定的规格，而芯片组对内存的支持通常也仅支持特定规格，所以，当你选择 Intel 的主机板芯片组时，那就不能使用 AMD 的 CPU 啰！ 这在购买的时候要特别小心您主机板上芯片组所能支持的规格喔！ 使用者最容易搞错的就是这里了！大家总是认为 DDR 最好，所以拼命也要买 DDR，但是，如果你主机板芯片组本来就不支持 DDR 内存的话，那你买来的 DDR 是找不到地方插的！所以， 如果您想要升级你的系统时，请特别留意你的主机板芯片组是否有支持喔！

- I/O 地址与 IRQ 岔断： 既然主机板是负责各个计算机系统组件之间的沟通的，但是计算机的东西又太多了， 又有输出输入、又有不同的储存装置，主机板芯片组怎么知道如何负责沟通呐？ 这个时候就需要用到所谓的 I/O 与 IRQ 啰！I/O 有点类似门牌地址啦， 每个装置都有他自己的地址，一般来说，不能有两个装置使用同一个 I/O 地址， 否则系统就会不晓得该如何运作，例如，如果你家门牌与隔壁家的相同， 那么邮差怎么送信到你家啊？不过，万一还是造成不同的装置使用了同一个 I/O 而造成 I/O 冲突时，就需要手动的设定一下各个装置的 I/O 啰!而除了 I/O 地址之外，还有个 IRQ 岔断这个咚咚，如果 I/O 想成是门牌号码的话，那么 IRQ 就可以想成是各个门牌连接到邮件中心（ CPU ）的专门路径啰！IRQ 可以用来沟通 CPU 与各个装置啦！目前 IRQ 只有 15 个，如果你的周边接口太多时， 可能就会不够用，这个时候你可以选择将一些没有用到的周边接口关掉，以空出一些 IRQ 来给真正需要使用的接口喔！当然，也有所谓的 sharing IRQ 的技术就是了！
- BIOS ：BIOS 是 Basic Input/Output System 的缩写，刚刚上面我们提到了很多的输出与输入以及 I/O, IRQ 等等的咚咚， 你要如何设定呢？可以透过操作系统，也可以透过主机板提供的 BIOS 功能来设定喔！当你开机的时候，屏幕上不是会出现一些版本的讯息吗？那就是 BIOS 的设定啰！你可以在开机的时候按下 DEL 按键，以设定开机顺序、 I/O ，以及 IRQ 等等！
- 其它输出输入接口： 目前主机板上面通常会内建一些基本的接口，这些接口通常是在主机机壳的后面，例如：

    - PS2 界面：这是目前最常见的键盘与鼠标的接口， 在插孔的地方是圆形的，这种接口速度上面会比较好一些，但是最麻烦的地方在， p 果你的键盘与鼠标松脱了，通常只能以重新开机来再次驱动键盘或鼠标啰；
    - 九针串行端口：这个是以前用来接鼠标的接口， 常常被称为 com1 ；
    - 25 针并列埠：这通常用来连接打印机的接口，通常称为 LPT1, LPT2...；
    - 声音输出、输入与麦克风： 这个是一些圆形的插孔，而必须你的主机板上面有内建音效芯片时，才会有这三个东西；
    - USB 界面：目前相当流行的一个接口， 支持随插即用，目前已经推出到 USB 2.0 了，这个规格改变了速度上的问题， 目前 USB 2.0 的速度已经足够了 (480 Mbps)，相当的快速！不像之前 1.xx 版时 (12 Mbps)， copy 一些数据到 USB 硬盘时，会吐血....

- 电源供应器 (Power)：

除了上面这些组件之外，其实还有一个很重要的组件也要来谈一谈，那就是电源供应器。 在您的机壳内，有个大大的铁盒子，上头有很多电源线会跑出来， 那就是电源供应器了。我们的 CPU/RAM/主机板/硬盘 等等都需要用电， 而近来的计算机组件耗电量越来越高，以前很古早的 230W 电源已经不够用了！ 最近您要安装新的主机时，要记得，电源供应器至少也要 300W 以上才够你的主机使用， 有些特殊的主机，还会要求至少要 400~500W 以上的电源呢！

电源供应器的价差又更大了！贵一点的 300W 可以到 4000 NT，便宜一点的 300W ， 只要 500 NT 不到！怎么差这么多？没错～因为 Power 的用料不同， 电源供应的稳定度也会差很多，差一点的电源供应器，甚至是造成计算机不稳定的元凶呢！ 所以，尽量不要使用太差的电源供应器喔！

- 关于速度/稳定度的问题：

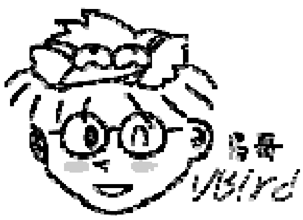
对于速度来说的话，由刚刚我们看到的芯片组所负责沟通的工作来看， 您就应该晓得啦！速度的快慢与『整体系统的最慢的那个设备有关！』 ，如果你是使用最快速的 P4 ，使用最快的 DDR 内存， 但是配上一个慢慢的过时显示卡，那么整体的速度效能将会卡在那个显示卡上面喔！ 很重要的呢！所以，在购买整套系统时，请特别留意需要全部的接口都考虑进去喔！ 尤其是当您想要升级时，要特别注意这个问题，并非所有的旧的设备都适合继续使用的。

除此之外，到底那个组件特别容易造成系统的不稳定呢？有几个常见的系统不稳定的状态是：

- 系统超频：这个行为很不好！不要这么做！
- 电源供应器的电源不稳定：这也是个很严重的问题，当您测试完所有的组件都没有啥大问题时，记得测试一下电源供应器的稳定度！
- 内存无法负荷：现在的 DDR 内存质量差很多，差一点的内存， 可能会造成您的主机在忙碌的工作时，产生不稳定或当机的现象喔！
- 系统过热：『热』是造成电子零件运作不良的主因之一，如果您的主机在夏天容易当机， 冬天却还好，那么考虑一下，加几个风扇吧！有助于机壳内的散热， 系统会比较稳定喔！『 这个问题也是很常见的系统当机的元凶！ 』

这些咚咚就是系统所必备的一些组件了！当然啦，还有光驱、鼠标、键盘我们没有介绍， 因为那个东西比较简单啦！只是要注意的是，他与主机板提供的连接接口是否相同呢？ 如果不同的话，自然就无法连接啰！例如你拿 PS2 的接头要去接九针串行端口的插槽，试问，可以连接吗？？ 所以，买接口设备的时候，要考虑到整体性喔！

Tips:
事实上，要了解每个硬件的详细架构与构造是很难的！这里鸟哥仅是列出一些比较基本的概念而已。 另外，要知道某个硬件的制造商是哪间公司？可以看该硬件上面的信息。 举例来说，主机板上面都会列出这个主机板的开发商与主机板的型号，知道这两个信息， 就可以找到驱动程序了。另外，显示卡上面有个小小的芯片，上面也会列出显示卡厂商与芯片信息喔！



选择与 Linux 搭配的主机配备

认识了主要的计算机系统硬件之后，接下来就得知道， 那么你的硬件需要怎样的等级才能符合你的操作系统需求呢？刚刚我们也提到了， 不同的主机服务需要的主机等级是不相同的！基本上，如果你的 Linux 是做为终端机之用 (也就是当作你的工作机，并不对 Internet 提供其它服务)，并且也不玩 X-Window 的话，那么由于 Linux 所需要的硬件资源是很低的，只要你有 Pentun-133 以上等级的机器就可以跑得很顺畅啰！所以，若你手上有升级后要汰换的设备， 千万别急着丢掉，可以尝试着组装一下，然后来玩玩 Linux 呀！

但是，万一您的 Linux 机器是预计用来作为您公司内部的 mail server 或者是您学校的 Web server, proxy server 时，或者是要玩 X-Window 的话， 那么你就必须要选择高档一点的计算机配备了，尤其是 RAM 的大小、显示卡的内存容量与硬盘的空间容量！

另外，由于 Linux 还没有茁壮到大部分的桌上型计算机的操作系统都用他，因此， 某些特殊硬件对于 Linux 之支持度，就有点不足了。 举例来说，最新的显示卡可能就无法被预设的 Linux 核心捉到。 不过，其实这并不是 Linux 的问题～怎么说呢？因为驱动程序都是由该硬件的开发商开发的， 而不是 Linux 核心工作小组应该要开发的喔！所以啊， 如果您买了一个显示卡，却无法被你的 Linux 侦测到，那么您应该要前往这个显示卡开发商的网站， 去反应，或者去下载给 Linux/Windows 或其它操作系统使用的驱动程序才对喔！

此外， Linux 开发商在释出 Linux 之前，都会针对该版所预设可以支持的硬件做说明， 因此，您除了可以在 Linux 的 Howto 去查询外，也可以到各个相关的 Linux distributions 网站去查询呢！

- Linux 的硬件中文 HowTo： http://www.linux.org.tw/CLDP/HOWTO/hardware.html#hardware
- Mandriva 的硬件支持： http://www.linux-mandrake.com/en/hardware.php3
- Red Hat 的硬件支持： http://hardware.redhat.com/hcl/?pagename=hcl
- SuSE 的硬件支持： http://hardwaredb.suse.de/index.php?LANG=en_UK
- Linux 对 Printer 的支援： http://www.linuxprinting.org/
- Linux 对 Notebook 计算机的支持： http://www.linux-laptop.net/
- 显示卡对 XFree86/Xorg 的支持： http://www.linuxhardware.org/

底下我们稍微谈一下 Linux 至少所需要的硬件配备是如何吧！假设一台 Linux 主机，他主要的功能是用来作为 NAT 主机，所谓的 NAT 主机也就是类似『IP 分享器』 的功能，而且用这台 NAT 主机的 PC 数并不多，那你只需要 Pentun-166, 32MB RAM, 及一块不太特殊的显示卡及网络卡也就够了！ 当然，硬件的需求与你服务的对象多寡是有相当的相关性的！在这个一般家庭的 NAT 主机的环境下，你所需要的硬件大致的需求如下：

- CPU：Pentun-166 以上等级就可以了。不过建议使用 K6-2 300 以上等级的 CPU，当然了， CPU 的等级与你旧有的主机板兼容程度是有相关性的；
- RAM：至少 32MB 以上。其实除了 CPU 之外， 在 Linux 系统中最重要的应该是内存的大小了，因为如果你的服务开得太多， 而你的内存不够大，势必要使用类似 Windows 的 『虚拟内存 』的东西 (在 Linux 当中称为 Swap) ，这个 Swap 是使用硬盘的空间来仿真内存的存取型态，所以，你可以知道，在内存中跑的数据却以速度较慢的硬盘来跑， 呵呵！这东西可操硬盘的紧！所以虽然内存最低的需求是 32MB 就可以了，不过强烈建议最好是有 64MB 以上比较好，尤其是如果你还要玩 X-Window 的话！ (注：目前新出版的 Linux distribution 当中，由于提供的服务越来越多，且 X-Window 接口越做越好，所以对于内存的要求，实际上也越来越高！事实上，最好要求您的 Linux 具有 128 MB 以上的内存，不过，如果您跟鸟哥 一样不碰 X-Window 的话，那么使用 64 MB 就已经吓吓叫了！)
- Hard disk：最好有 2GB 以上。 当然是越大越好，最好至少为 3GB 的硬盘！（注：同样的，目前的 Linux 提供的资料太多了！所以某些出版商提供的 Linux 在选择完整安装之后，硬盘竟然占用了 4.5 GB 左右的空间，相当的可怕！不过，如果您已经学会了 Linux 的话， 那么事实上，透过选择的套件内容，将不需要用到这么多硬盘空间，尤其不玩 X-Window 的话，硬盘空间几乎可以减少一半以上。）
- VGA (显示卡)：如果是旧设备的话，最好是 S3 早期的显示卡。 Linux 对于最新的显示卡支持的并不是很足够，而且通常鸟哥 是建议人家使用淘汰的零件当主机使用，并且如果你又不玩 X-Window 的话，一块 1MB 内存的 S3-775 显示卡就够了！重要的是， Linux 对于 S3 旧的 VGA 卡 (如 Virge 系列) 支持的程度相当的成熟，所以我推荐他!然而，如果您想要将 X-Window 建置在您的 Linux 机器上面，那么最好是『一定要有 8 MB 以上的显示卡内存』，否则光是等待的时间，会磨尽您原本具有的耐心指数….
- Network Card (网络卡) ：一块极其普通的 10/100 MB 的网卡就可以了，建议用具有 RTL8139 或者是 NE2000 兼容的芯片的网卡，因为 Linux 本身就有支持，不用再额外加挂驱动程序！强烈的建议使用 Realtek RTL8139 芯片的 PCI 接口网络卡，便宜又蛮好用的！不过，还是得提醒一下，如果您的 Linux 是用来架设大流量的网站时，那么好一点的网络卡将是不能节省的花费！如果能够使用 Intel 或是 3Com 的网络卡，那将是不错的选择呀！

- 光盘、软盘、键盘与鼠标：不要太旧的就可以了。 基本上除了键盘之外，其它的装置都是非必备的，以鸟哥为例， 在安装 Linux 的时候先拿别部计算机的光驱、软盘机与鼠标来安装，等到安装完毕之后， 关机，将所有的装置拔掉，只要剩下硬盘与电源供应器就可以啦！等到所有的设定都完成之后， 连屏幕都可以搬走了！剩下的 Linux 会自动搞定！因为通常服务器这东西最需要的就是稳定， 而稳定的最理想状态就是平时没事不要去动他是最好的！

不过，请千万注意了，上面提到的是『规模很小的主机系统』可以这样玩！如果是『企业内部的 Linux 主机』，呵呵，可能就要做修正啰！例如某些学校内部架设的 Proxy 系统， 由于服务的机器数非常的大，所以建议至少需要：

- CPU 等级至少需要 P-III 500 以上；
- RAM 最重要，最好至少 512 MB 以上，越大越好；
- 网络卡最好可以选择较佳一些的，例如 Intel 或 3COM 的！
- 硬盘至少需要数十 GB 以上的，分割成多槽， Proxy 执行效率较好；
- 其它的就随意啦！

所以啰！不同规模的服务器，他的硬件要求等级也就会不相同！除此之外，不同的 Linux distribution 对于硬件的要求也不一样！举例来说，在 Open Linux 的 server 3.1.1 就『严格要求』您的系统必须是 i686 （ 也就是 PII 等级以上的 CPU ），所以，您必须要针对您即将安装的 Linux 所需要的硬件需求进行了解呢。

Tips:
一般来说，目前 (2005/06) 的入门计算机机种，至少都会有 P-4 2G 以上， RAM 有 512MB ，显示卡内存也有 64MB 以上，所以，如果您是新购置的计算机，那么该计算机用来作为 Linux 的练习机，而且加装 X Window 系统， 肯定是可以跑的吓吓叫的啦！ ^_^



底下鸟哥针对一般您可以会接触到的计算机主机的用途与相关硬件配备的基本要求来说明一下好了：

- 一般小型主机且不含 X Window 系统：
    - 用途：家庭用 NAT 主机或小型企业之非图形接口小型主机。
    - CPU：大于 Pentun 133 以上等级即可。
    - RAM：至少 32MB ，不过还是大于 64MB 以上比较妥当！
    - 网络卡：一般的 10/100 Mbps 即可应付。
    - 显示卡：随便！只要能够被 Linux 捉到即可，例如 S3 或 Sis 6326
    - 硬盘：2GB 以上即可！

- 桌上型 Linux 系统/含 X Window ：
    - 用途：Linux 的练习机或 Office 工作机。
    - CPU：最好等级高一点，例如 P-III 或 K7 以上等级。
    - RAM：一定要大于 256MB 比较好！否则容易有停顿的现象。
    - 网络卡：普通的 10/100 Mbps 就好了！
    - 显示卡：使用 32MB 以上内存的显示卡！
    - 硬盘：越大越好，最好有 20GB 。

- 中型以上 Linux 服务器：
    - 用途：中小型企业/学校单位的 FTP/mail/WWW 等网络服务主机。
    - CPU：最好等级高一点，例如 P4 或 K7 以上等级。甚至可以考虑使用双 CPU 系统。
    - RAM：最好能够大于 512MB 以上，大于 1GB 更好！
    - 网络卡：知名的 3Com 或 Intel 等厂牌，比较稳定效能较佳！注意，也可选购 10/100/1000 Mbps 的速度。
    - 显示卡：如果有使用到图形功能，则一张 64MB 内存的显示卡是需要的！
    - 硬盘：越大越好，如果可能的话，使用 SCSI 或者磁盘阵列，或者网络硬盘等等的系统架构， 能够具有更稳定安全的传输环境，更佳！

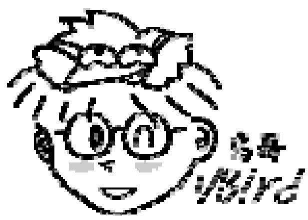
总之，鸟哥这里仅是提出一个方向，亦即是：如果您有因为升级而用不到的计算机主机， 千万不要急着丢掉，可以将他回收后，作为 Linux 的架设与练习之用！ 而如果您想要架设一部更稳定的 Linux Server ，那么，系统的整体搭配性、 整体运作的效率考虑，以及系统散热的问题等等，都需要加以考虑。在综合考虑之后， Linux Server 在中大型企业上，购买各硬件厂商已开发完成的硬件系统， 是一个很不错的选择！至少那些服务器主机都已经测试过搭配性，而且散热上一定比较没问题！

总之，如果是自己维护的一个小网站，考虑到经济因素，您可以自行组装一部主机来架设。 而如果是中、大型企业，那么主机的钱不要省～因为，省了这些钱，未来主机挂点时， 光是要找出那个组件出问题，或者是系统过热的问题，会气死人ㄟ！ 而且，要注意的就是未来你的 Linux 主机规划的『用途』来决定你的 Linux 主机硬件配备喔！ 相当的重要呢！

各硬件装置在 Linux 中的代号

了解了硬件之后，接着下来得了解一下个硬件在 Linux 当中所扮演的角色啰！ 在 Linux 系统当中，每个装置都被当成一个档案来对待！ 举例来说，硬盘的文件名称即为 /dev/hd[a-d] ，其中， 括号内的字母为 a-d 当中的任何一个，亦即由 /dev/hda, /dev/hdb, /dev/hdc, 及 /dev/hdd 这四个档案的意思 (注：这种型式的表示法在后面的章节当中会使用得很频繁，请特别留意)。 那么光驱与软盘呢？分别是 /dev/cdrom, /dev/fd0 啰！ 好了，其它的接口设备呢？底下列出几个常见的装置与其在 Linux 当中的代号啰：

> Tips:
> 先提出来强调一下，在 Linux 这个系统当中，几乎所有的硬件装置代号档案都在 /dev 这个目录当中，所以您会看到 /dev/hda, /dev/cdrom 等等～



装置	装置在 Linux 内的代号
IDE 硬盘机	/dev/hd[a-d]
SCSI 硬盘机	/dev/sd[a-p]
USB 随身碟	/dev/sd[a-p](与 SCSI 硬盘一样)

CDROM	/dev/cdrom
软盘机	/dev/fd[0-1]
打印机	/dev/lp[0-2]
鼠标	/dev/mouse
磁带机	/dev/ht0 (IDE)或 /dev/st0 (SCSI 界面)

需要特别留意的是硬盘机(不论是 IDE/SCSI/USB 都一样)，每个磁盘驱动器的磁盘分割 (partition) 不同时，其磁盘代号还会改变呢！ 关于 硬盘机的分割与配置将在安装 Linux 时再提及。此外，您会发现怎么档案开头都是 /dev 呢？呵呵！那个咚咚就是我们放置装置档案的目录啦！而需要特别注意的是磁带机的代号，在某些不同的 distribution 当中可能会发现不一样的代号，需要稍微留意。总之，你得先背一下 IDE 硬盘的代号就是了！其它的，用的到再来背吧！

安装 Linux 前的规划

操作系统与硬件相关性是很高的，我们刚刚也才谈过 x86 这个个人计算机架构的各硬件组件， 也大略的介绍了一些选购的注意事项，再来是什么？呵呵！再来则是需要知道那我应该要安装那个版本的 Linux ？在安装的过程当中，我应该要如何将我的硬盘进行分割？还有，我应该要如何选择要安装的 Linux 套件(软件)？ 因为每个不同的 Linux 开发商在开发他们的 Linux 时，着眼点都不同，所以当然就要选择比较适合您的版本啰。 至于硬盘分割，那本来就是一件很重要的事情～不论是在那个操作系统当中啊！而 Linux 的软件众多， 没有必要每个都安装在您的主机上面的说！呵呵！底下我们就分别来谈一谈这些东西啰！

选择适当的 distributions

就如同前面几个章节提到的，每个版本的 Linux 都是使用 http://www.kernel.org 所发展的核心， 都遵循 LSB 与 FHS 等等的架构，所以差异性其实不大啦！不过，每个 Linux distributions 在发展的时候，都有锁定他们的用户群，因此，在『预设的情况下』，每个版本都有比较特别适合的使用群。 举例来说，Ubuntu (http://www.ubuntulinux.org/) 就比较适合桌上型计算机使用， 因为他的 X Window 整合得很好。 Red Hat Enterprise Linux 与 SuSE Enterprise Linux Server 就比较适合企业的 Linux 主机，因为他们的系统服务整合的比较好。

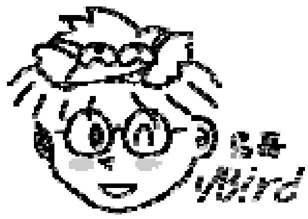
但是，上面提到的都是『预设情况下』的使用状态，事实上，因为每个 linux distributions 差异性不大，所以，您当然可以随意选择一个 distributions 来加以改造， 以符合您自己的喜好的环境啊！ 不过，要注意的是，由于近期以来，网络的怪客 (Cracker) 很多， 造成我们主机的被入侵的危险性大增！因此，您要选择的 distributions 的标准之一， 就是：『选择比较新的 distribution 为宜！』这是因为比较新的版本他在 持续维护套件的安全性上，比较长，可以让您的系统比较安稳一点。而且，比较新的 distributions ， 他在新硬件的支持上面，当然也会比较好啰！这样可以了解吗？

您可以在 Linux 是什么 那个章节当中介绍的 Linux distributions 选择适当的 distribution 去下载来安装，不过，那些网站大多是国外的网站，下载时间会较久。 这里介绍国内的学术网络，例如义守大学的 FTP 网站： http://ftp.isu.edu.tw/pub/Linux/ 去下载最新的安装光盘版本。 鸟哥在这里给您建议，

以台湾目前而言，使用者群使用 Fedora 及 Mandriva 还不少， 这表示使用这两个版本若发生问题时，应该可以得到比较多的参考数据，所以，您可以选择这两个套件其中之一， 来开始练习您的 Linux 啊！

另外，您也可以选择国外的一个提供几乎全部 Linux distributions 的网站：http://www.linuxiso.org/ 来下载。 要注意的是，以义守大学的 FTP 提供的 FC4 (Fedora Core Release 4) 为例，他的下载点：http://ftp.isu.edu.tw/pub/Linux/Fedora/linux/core/4/i386/iso/ 里面有好多档案， 每个档案都很大！这是因为，那些档案都是映象档(image file)，还必须要烧录成为光盘后， 才可以使用。而您也会看到里头有 i386 及 i386-SRPMS 的档案，那个 SRPMS 的档案是含有原始码的， 目前我们使用不到，所以可以略过不下载，只要下载 FC4-i386-disc[1-4].iso 即可。 (注：提供 Linux distributions 下载的网站很多，您可以到各大专院校的 BBS 站的精华区去搜寻一番！)

> Tips:
> 要注意的是，因为 images 档案实在太大了，通常是 600MB-700MB 之间， 这么大的档案使用浏览器的接口 (如 IE 或 Firefox) 来下载可能会有问题， 例如断线啦等等的。所以这里请您以 FTP 的软件 (例如 cuteftp 等等的) 来下载， 这样不但可以避免断线，也拥有续传的功能，而且档案取得也会比较完整。



主机的服务规划与硬件的关系

前面已经提过，由于主机的服务目的不同，所需要的硬件等级与配备自然也就不一样！ 底下鸟哥稍微提一提每种服务可能会需要的硬件配备规划，当然，还是得提醒， 每个朋友的需求都不一样，所以设计您的主机之前，请先针对自己的需求进行考虑。而， 如果您不知道自己的考虑为何，那么就先拿一部普通的计算机来玩一玩吧！不过要记得！ 不要将重要数据放在练习用的 Linux 主机上面。

- 打造 Windows 与 Linux 共存的环境：

在某些情况之下，你可能会想要在『一部主机上面安装 两套以上的操作系统』，举例来说：

一、我的环境里面仅能允许我拥有一部主机，不论是经济问题还是空间问题～
二、因为目前各主要硬件还是针对 Windows 进行驱动程序的开发，我想要同时保有 Windows 操作系统与 Linux 操作系统，以确定在 Linux 底下的硬件应该使用那个 I/O port 或者是 IRQ 的分配等等；
三、我的工作需要同时使用到 Windows 与 Linux 操作系统。

果真如此的话，那么您就可能会需要使用到所谓的 『多重开机』选单系统了 ！所谓的多重开机选单，就是在系统开机时， 可以让您选择进入哪一种操作系统的程序。因为如此，所以，您就可以在一部主机 上面安装两套操作系统在不同的磁盘分割槽内，此时您就能够以一部主机来操弄两个操作系统了。

> Tips:
> 一般来说，您还可以在 Windows 操作系统上面安装 VMware 之类的软件， 让您可以在 Windows 系统上面使用 Linux 系统，就是两个操作系统同时启动！ 不过，那样的环境比较复杂，尤其很多硬件都是仿真的，会让新手很难理解系统控制原理。 基本上，鸟哥很不建议您使用这样的方式来学习 Linux 喔！



举例来说，假设您想要同时安装 Windows XP 与 Linux 在您的工作主机上面，那你必须先安装 Windows XP 再灌 Linux 系统就可以了！当然啰，如果你先安装了 Linux 再安装 Windows 系统呢？还能不能成功的制作多重开机？当然可以啦！不过， 你就需要学会知道什么是『多重开机』的概念，这部分我们会在后面再

继续谈，不要着急喔！ 基本上，多重开机涉及硬盘规划的问题，如果你的硬盘有 6GB ，那你可以先以 DOS 的 Fdisk 或其它的分割程序如 SPFdisk http://spfdisk.sourceforge.net/ 进行硬盘的划分。仅割出 Windows 要的扇区就好。例如你要分 1GB 给 windows ，那以 Fdisk 分割一个 1GB 的主分割就好了！其它的等 Linux 灌的时候再弄就可以啦！这部分会在后面再提到！

再来提到您的 Linux 主机系统，我这里要跟大家报告的是， 如果您是使用较为老旧的计算机来做为主机的处理，并且他上面可能预计会安装 mail, WWW 等服务器软件，因此需要全天、全年开机的，所以安装 Windows 与 Linux 共存的环境是可以，但是请将 Windows 的硬盘规划的小一点！ 好让您的 Linux 主机可以有更多的空间提供更完善的服务。 好了！现在来说说你需要的主机服务有哪些呢？ 一般而言，对于非企业或者是小型企业或者是学校单位，通常你需要的服务有底下这几个：

- NAT (类似 IP 分享器的功能)：

如果您是一般小型企业，或者是一般的中小学学校， 那么贵单位对外的联机应该通常是：『申请一个固定制的 IP ，然后透过 IP 分享器 (IP sharing) 来达到全校的计算机皆可连上 Internet 的联机机制』吧！咦！要连上 Internet 不是需要公共 IP 吗(Public IP)，那每部计算机不是都需要一个 IP 吗？那么您只有申请一个 Public IP ，其它计算机的 IP 要怎么设定呢？早在当初规划这个 IPv4 协议时(就是目前的 IP 设定啰！)，就考虑到可能的 IP 不足啦！此时，就有专门给内部网域设定用的 Private IP 了(或者称为私有 IP 或保留 IP) ，需要注意的是，这些 Private IP 都不能直接与 Internet 上面的 Public IP 互相沟通！

那怎么我学校内部的计算机还是可以透过 IP 分享器连出去呢？这就是所谓的 NAT(Network Address Translation)功能啦！当内部计算机要连接上 Internet 时，需要通过 NAT 的技术，将你内部计算机的数据封包中，关于 IP 的设定都设定成 NAT 主机的公共 IP ，然后才传送出去 Internet ，如此一来，你的内部计算机虽然是使用私有 IP ,但是在联机上 Internet 时,就可以透过 NAT 主机的 NAT 技术,将 IP 来源给改了改！哈哈！如此一来，就可以向 Internet 要求数据啰！这部分我们在 网络基础篇 会再提及的！通常使用旧计算机来做为主机时，最大的效用就是用来作为 NAT 了！若你的主机仅单纯提供 NAT 服务，那么在这个服务当中，比较重要的就属网络卡而已！其它的 CPU、RAM、硬盘 等硬件的影响相对就小了相当多！

- SAMBA (类似网络上的芳邻功能)：

在 Windows 里面可以很轻易的就以『网络上的芳邻』来分享彼此的档案数据，那么 Linux 要如何与 Windows 分享呢？呵呵！使用 SAMBA 就可以啦！ 这也是最普遍的 file server （档案服务器）。 由于分享的数据量可能较大，那么对于系统的网络卡与硬盘的大小及速度就比较重要， 如果您针对不同的使用者提供档案服务器功能，那么 /home 可以考虑独立出来，并且加大容量。

- Mail (邮件服务器)：

Linux 一安装完毕就已经提供了 Sendmail 或 Postfix 的邮件服务！ 由于我们如果向外面的公司申请免费的 E-Mail 信箱，了不起容量大致上到 20 MB ， 但是，要知道有时候我们一不小心就会让邮件容量超过了 20 MB，这样一来，呵呵，您的免费信箱就爆了！真抱歉....但是，如果你自己架设一个 mail server 呢？哈哈！哪么你的信箱就可以到达几 GB 这么大！很过瘾吧！在 mail server 上面，重要的也是硬盘容量与网络卡速度，在此情境中，也可以将 /var 独立出来，并加大容量。

- Web (WWW 服务器)：

WWW 几乎是每个主机上面都会安装的一个套件了！当然，要推销你自己的话，那么 WWW 服务器是绝对不会被你忘掉的！在 Web server 上面，CPU 的等级有时候不能太低，而最重要的则是 RAM 了！要增加 WWW 系统的稳定度，提升 RAM 是一个不错的考虑。

- DHCP (提供自动取得 IP 的功能)：

NAT 搞定之后，要晓得的是，你的 Client （客户端）每一部都需要经过设定才能上网 (刚刚提到的私有 IP 的概念！) ！阿！好麻烦！那么使用 DHCP 就可以改善这个问题啰！呵呵！ Client 端都不必设定任何咚咚，马上将可以上网了！快乐吧！ 这个咚咚的硬件要求可以不必很高啰。

- Proxy (代理服务器)：

这也是常常会安装的一个服务器软件，尤其像中小学校的频宽较不足的环境下， Proxy 将可有效的解决频宽不足的问题！当然，你也可以在家里内部安装一个 Proxy 喔！但是，这个服务器的硬件要求可以说是相对而言最高的，他不但需要较强有力的 CPU 来运作，对于硬盘的速度与容量要求也很高！ 自然，既然提供了网络服务，网络卡则是重要的一环！

- FTP：

FTP 的功能是真的很好啦！但是对于 拨接制 ADSL 使用者来说，架设 FTP 实在是一件不智的事情！因为对你的频宽影响太大了！ 鸟哥 相当不建议架设 FTP 的啦！尤其安全性上面也很伤脑筋！ 对于 FTP 则是您的硬盘容量与网络卡好坏相关性较高。

大致上我们会安装的服务器软件就是这一些啰！假设您需要 NAT 的服务， 那么通常会建议安装『两块网络卡』在您的主机上面，因为可以顺便解决您内部计算机的安全问题！假如您需要 mail 与 Web 服务器，那么就建议申请 DNS 或者是直接申请免费的动态 DNS 系统的 domain name 啰！如果您需要 Proxy 的服务，那么在当初设计硬盘规划的时候， 就要小心硬盘的分割了，因为不同的切割方式会使得您的 Proxy 效能有差异！

当然啦，还是那句老话，目前我们这本书里面谈论的，还是以 Linux 基础为主，鸟哥也希望您先了解 Linux 的相关主机操作技巧，其它的架站，未来再谈吧！ 而上面列出的各项服务，仅是提供给您，如果想要架设某种网络服务的主机时，您应该如何规划主机比较好！

主机硬盘的主要规划

系统对于硬盘的需求跟刚刚提到的主机开放的服务有关，那么除了这点之外， 还有没有其它的注意事项呢？当然有，那就是数据的分类与安全性的考虑。 常常会发现网络上有些朋友在问『我的 Linux 主机因为跳电的关系，造成不正常的关机，结果导致无法开机，这该如何是好？』呵呵，幸运一点的可以使用 fsck 来解决硬盘的问题，麻烦一点的可能还需要重新安装 Linux 呢！伤脑筋吧！另外，由于 Linux 是多人多任务的环境，因此很可能上面已经有很多人的数据在其中了， 如果需要重新安装的话，光是搬移与备份数据就会疯掉了！所以硬盘的分割考虑是相当重要的！

同时，硬盘的规划对于 Linux 新鲜人而言， 那将是造成您『头疼』的主要凶手之一！因为硬盘的分割技巧需要对于 Linux 档案结构有相当程度的认知之后才能够做比较完善的规划的！ 所以在未来的几个章节当中，鸟哥将会着重在这方面的探讨， 这可是相当重要的入门知识呢！因为如此，所以特别建议 Linux 新鲜人先只切两个扇区就好，分别是根目录 / 与 Swap ！ 无论如何，底下还是说明一下基本硬盘分割的模式吧！

- 最简单的切割方法： Linux 安装的过程中，至少要有两个 partition 才行，一个是『 / 』，另一个则是虚拟内存『 Swap 』，如果你的硬盘很小（例如小于 1GB 的小硬盘），那么使用这个分割的方法会比较好！但是，保证是比较不保险的切割方式啦；
- 稍微麻烦一点的方式：在预设的情况下，由于 Linux 的操作系统都是摆在 /usr/ 当中，所以啰，你可以将这个部分切割的大一点，另外，由于使用者的信息都是在 /home 底下，因此这个也可以大一些，而 /var 底下是记录所有预设服务器的登录档，且 mail 与 WWW 预设的路径也在 /var 底下，因此这个空间可以加大一些喔！所以，需要的目录就有：
    - /
    - /usr
    - /home
    - /var
    - Swap

以鸟哥为例，通常我会希望我的邮件主机大一些，因此我的 /var 通常会给个数 GB 的大小，如此一来就可以不担心会有邮件空间不足的情况了！另外，由于我开放 SAMBA 服务，因此提供每个研究室内人员的数据备份空间，所以啰， /home 所开放的空间也很大！至于 /usr/ 的空间，大概只要给 2-3 GB 即可！凡此种种均与您当初预计的主机服务有关！ 因此，请特别注意您的服务项目！然后才来进行硬盘的规划

鸟哥说：关于练习机的安装建议

- 关于硬件方面

一般来说，对于学习 Linux 这个操作系统，最麻烦也最重要的地方，就是一开始的安装了。 很多朋友都是一开始安装 Linux 就遭遇到困扰，导致没有兴致再继续往下来学习 Linux 。 造成这样的安装困扰，很多都是因为朋友们只有一部主机，而在还没有了解到磁盘档案系统的运作，就贸然的进行多重开机的规划，导致系统不小心被损毁，进而不想继续使用 Linux 。 另外则有一些朋友是利用类似 VMWare 的软件来学习 Linux 。但是因为 VMWare 里面的硬件很多都是仿真的， 造成朋友们不知道该如何分辨问题发生的所在，而无法继续学习。

有鉴于此，因此，鸟哥『强烈的建议您，务必拥有一台主机， 而且内含一颗仅有 Linux 操作系统的硬盘』，以鸟哥自己为例， 我的主机上面有一个抽取式硬盘盒，而我有两颗分离的硬盘， 分别安装 Windows 与 Linux 系统，要使用 Linux 时，就插入 Linux 硬盘， 使用 Windows 时，就插入 Windows 硬盘，如此一来，主机很单纯， 而抽换也很快速，不需要对机壳拆拆装装的，很方便！提供给您做为参考。

- 关于硬盘分割方面

此外，在硬盘的分割方面，鸟哥也建议新手们，先暂时以 / 及 swap 两个分割即可， 而且，还要预留一个未分割的空间喔！因为我们是练习机，暂时不会提供网络服务， 所以只要有 / 及 Swap 提供给我们进行安装 Linux 的空间即可。不过，我们未来会针对系统的磁盘部分进行分割的练习以及磁盘配额 (quota) 的练习， 因此，预留一个磁盘空间是必须要的！

举例来说，如果您有一个 20GB 的硬盘，那么建议您，分 15 GB 给 / 来安装 Linux ， 512 MB 给 Swap ，另外的 4GB 左右不要分割，先保留下来，未来我们可以继续来练习喔！^_^

- 关于软件方面

另一个容易发现问题的地方，在于使用者常常会找不到某些指令，导致无法按照书上的说明去执行某些指令。 因为无法执行指令，所以就会一直给他放在那边，不会继续往下学习啊！真是可惜！ 为什么会找不

到指令呢？很简单啊！就是因为没有安装该套件(软件)啊！ 所以，『强烈的建议新手，务必将所有的套件都给他安装上去！』 也就是选择『安装所有套件』就是了。

当然啦，上面提到的都是针对『练习机』而言喔！如果是您自己预计要上线的 Linux 主机， 那就不建议按照上面的说明安装了！切记切记！

鸟哥的两个实际案例

这里说一下鸟哥曾经规划过的两个范例，要先声明的，鸟哥的范例不见得是最好的， 因为每个人的考虑不同，我只是提供相对可能较佳的方案喔！

案例一：一般家庭使用的小型 Linux 主机：

- 提供服务：提供家里的五部计算机 ADSL 联机分享、同时架设 NAT Server、Mail Server、 WWW Server、SAMBA 等服务。此外，为多重开机系统。
- 架设硬件：
    - CPU 使用 P-166；
    - 内存大小为 64MB 的 RAM；
    - 网络卡为 螃蟹卡；
    - 硬盘机容量为 3.2 GB；
    - 显示卡选择 S3 Virge VGA。
    - 安装完毕之后拔掉 CD-ROM、鼠标、键盘、屏幕等等配备！只剩下网络线及电源线跟主机连接！
- 硬盘切割：
    - 提供 500 MB 给 Windows 98；
    - 1GB 给 /var (特别针对邮件设定)；
    - 100 MB 给 Swap；
    - 剩下的空间都给 /

案例二：提供约 100 部以上 PC 的 Proxy 主机设定：

- 提供服务：提供整个单位的 Proxy 服务器服务，同时提供单位内相关人员的数值模式仿真（这个模式很耗系统资源！）。
- 架设硬件：
    - 使用双 CPU 架构（因为需要大量的运算）；
    - 使用 GeForce 2 MX 显示卡（因为数值模式仿真完毕之后，需要将图标显示在屏幕上除错）；
    - 使用 30 GB 硬盘两颗（数值模式所需的储存、 Proxy 所需要的空间）；
    - 使用 3COM 网络卡（ Proxy 呦！）；
    - 使用 512 MB RAM。
- 硬盘切割：
    - 6 GB 给 Proxy （/proxy1, /proxy2, /proxy3 各占 2 GB）；
    - 1GB 给 Swap （数值模式需要）；
    - 5 GB 给 /；
    - 剩下的都给 /disk1 及 /disk2

在上面的案例中，案例一是属于小规模的主机系统，因此只要使用预计被淘汰的配备即可进行主机的架设！唯一可能需要购买的大概是网络卡吧！呵呵！；而在案例二中，由于我需要大量的数值运算，并且由于提供了很多计算机的 Proxy 服务，因此就需要较大的硬盘空间、与较佳的网络卡来搭配了！这些工作请先记得，因为下一章节在实际安装 Linux 之前，您得先进行主机的规划呀！

- 关于大硬盘

随着时代的演变，在 2005 年底的目前，呵呵～个人计算机上面的硬盘容量竟然都已经高达 160GB 以上了！这么大的硬盘用起来当然是很爽快的啦～不过，也有一些问题的～那就是～开机的问题～

因为 Linux 的开机程序『可能』会找不到 BIOS 提供的硬盘信息，这个不是 Linux 的问题， 而是 BIOS 本身无法支持这么大的硬盘的问题～啊！真困扰～虽然 Linux 的核心会『取代 BIOS』 而成功的侦测到大硬盘，不过，如果您将开机扇区安装在 > 1024 磁柱以后， 那么很可能你的 Linux 就会变成『可以安装，但是无法开机顺利使用』啦～

那怎么办？最简单的方法就是『将开机扇区规范在小于 1024 以内～』 即可！那怎么做到呢？很简单，在进行安装的时候，规划出三个扇区，分别是：

- /boot
- /
- swap

那个 /boot 只要给 100M Bytes 以内即可！而且 /boot 要放在整块硬盘的最前面！ 这部份您先有印象与概念即可，未来我们谈到开机流程时，会再加强说明的！ ^_^

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 请简略说明一部计算机主机里面，大概有哪些基本的硬件？

  一部计算机主机机壳内，一定都有主机板，主机板上面安插了 CPU、主存储器及显示卡等等； 另外还有排线与硬盘、光盘、软盘等等连接；主机机壳的背板则有输出输入的连接端口， 例如鼠标、键盘打印机等等；此外，还有一些 PCI 插槽，例如网络卡、声卡等等(有的是主机芯片组内建的)

- 一部计算机主机是否只要 CPU 够快，整体速度就会提高？

  不见得！一部计算机系统的速度与整体计算机系统的运作有关，每个组件皆会影响计算机的速度！这包括了内存、CPU、AGP 与显示卡速度，硬盘的速度以及其它相关的输入输出接口等等！ 所以，如果您的系统是升级的，那么还得必须要注意各个旧组件是否可以保留， 或者旧的可以用的组件必须要舍弃！

- 什么是 CPU 的外频与倍频？

  CPU 频率的计算当中，有所谓的外频与倍频，真正的频率需要将两者相乘才是！比较重要的是 CPU 的外频了！因为系统整体运作的频率便是依据这个外频来进行各个组件的沟通的！一般而言， 目

前比较流行的属于 133 这个外部频率，至于 PCI 则是这个频率的 1/4 倍，亦即是 33 MHz ， AGP 则是 66 （ 133/2 ），而，由于目前的技术越来越高超， CPU 可以透过特殊的技术来将外频调高为 133 的两倍，亦即是 266 ， RAM 也可以经过 DDR 的技术来将 133 加倍成为 266 ，这些技术都有助于速度上面的帮助！

- 什么是 I/O 地址与 IRQ 岔断？

  主机板是负责各个计算机系统组件之间的沟通的，但是计算机的东西又太多了， 又有输出输入、又有不同的储存装置，主机板芯片组怎么知道如何负责沟通吶？这个时候就需要用到所谓的 I/O 与 IRQ 啰！I/O 有点类似门牌地址啦，每个装置都有他自己的地址，一般来说， 不能有两个装置使用同一个 I/O 地址，否则系统就会不晓得该如何运作。不过， 万一还是造成不同的装置使用了同一个 I/O 而造成 I/O 冲突时，就需要手动的设定一下各个装置的 I/O 啰！而除了 I/O 地址之外，还有个 IRQ 岔断这个咚咚，如果 I/O 想成是门牌号码的话，那么 IRQ 就可以想成是各个门牌连接到邮件中心（ CPU ）的专门路径啰！IRQ 可以用来沟通 CPU 与各个装置啦！目前 IRQ 只有 15 个，如果你的周边接口太多时，可能就会不够用 ，这个时候你可以选择将一些没有用到的周边接口关掉，以空出一些 IRQ 来给真正需要使用的接口喔！

- Linux 对于硬件的要求需要的考虑为何？是否一定要很高的配备才能安装 Linux ？

  Linux 对于硬件的要求是因『服务种类、服务范围及主机的角色』而定的。例如一部专门用来运算数值解析的 Linux 运算工作站，需要比较强大的 CPU 与足够的 RAM 来进行工作，至于一般家庭用的仅用来做为 ADSL 宽带分享器的 Linux 主机，则只要 586 等级的计算机，甚至 486 系列的等级，就可以很顺利的运行 Linux 了。

- 一部好的主机在安装之前，最好先进行规划，哪些是必定需要注意的 Linux 主机规划事项？

  依据上一题的答案内容，我们知道 Linux 对于硬件的要求是『因地制宜』地！所以，要进行 Linux 的安装之前，一定需要规划 Linux 主机的定位与角色！因此， Linux 的主机是否开放网络服务？这部主机的未来规划中，是否需要进行大量的运算？这 部主机是否需要提供很大的硬盘容量来服务客户端的使用？这部主机预计开放的网络服务内容？ 等等，都是需要经过考虑的，尤其未来的『套件选择安装』上面，更需要依据这些规划来设定。

- 请写下目前您使用的个人计算机中，各项配备的主要等级与厂商或芯片组名称：

  主机板：
  CPU：
  内存大小：
  硬盘容量：
  显示卡：
  网络卡：

- 请写下下列配备中，在 Linux 的装置代号：
  IDE 硬盘：

CDROM：
打印机：
软盘机：
网络卡：

IDE 硬盘：/dev/hd[a-d]
CDROM：/dev/cdrom
打印机：/dev/lp[0-2]
软盘机：/dev/fd[0-1]
网络卡：/dev/eth[0-n]

- 如果您的系统常常当机，又找不到方法解决，您可以朝硬件的那个方向去搜寻？

  如果软件没有问题的话，那么当然发生当机的，可能就是硬件的问题了。 1.可以先检测系统有没有超频？ 2.再来则是查阅当系统运作时，系统的机壳内温度会不会过高？ 因为过高的温度常常会造成当机。 3.再者，检查一下 CPU 的温度，这也很重要。 4.再来，则是检查是否插了多条的内存，因为不同厂牌的内存混插很容易造成系统不稳定。 5.电源供应器是否合乎标准？这些都可以进行检测喔！

- 目前在个人计算机上面常见的显示卡接口有哪两个？

  AGP 与 PCI-Express 两种

- 目前在个人计算机上面常见的硬盘与主机板的连接接口有哪两个？

  有早期的 IDE 接口与最近的 SATA 接口，购买时要分的很清楚！

- 硬盘上面有所谓的跳针 (Jump) ，他是干嘛用的？

  由于一条 IDE 或 SATA 排线上面有两个装置的插入口，我们必须要藉由 Jump 来决定哪一个装置先被取用。目前有 Slave/Master/Cable select 等。

- 请上网查询，例如 Tom's hardware guide (http://www.big5.tomshardware.com/) 选择一款主机板芯片组，说明芯片组与 CPU/RAM/VGA/Hard disk 等等接口互相沟通的相关信息。

参考数据

- SPFdisk http://spfdisk.sourceforge.net/

一个简单的 SPFdisk 分割实例

最近更新日期：2005/06/12

磁盘分割是个很重要的学习知识！尤其是在您原本的硬盘空间不足了，或者是新增硬盘了， 或者是为了增加磁盘效能而必须要规划出比较适当大小的磁盘空间等等。 市面上很多工具可以让我们来进行磁盘的分割的，不过，都需要钱～ 当然，您也可以使用 Linux 的 fdisk 程序，不过，纯文字接口的方式，可能您也不容易学～ 相较之下，由台湾人自行开发的 spfdisk (special fdisk) 程序，不但纯中文接口， 使用图形接口的显示，耗用的系统资源又少！还可以作为开机管理程序！太完美了！赶紧来看看！

什么是硬盘分割(Partition)

在开始进行 Linux 之前，应该有很多的工作要做的！最重要的就如同前面『Linux 主机规划』当中说的，要如何规划硬盘呢？！到底要如何分割硬盘才好！是要将 swap （虚拟内存）规划的大一点比较好？或者是只要一个根目录就可以了呢？另外，如果我的硬盘上面已经有 Windows 系统，我又不想要将 Windows 杀掉，想使用多重开机来安装我的多个操作系统，那要怎么做呢？！况且，由于 DOS 的 fdisk 不认识 Linux 的档案格式，那么我要如何将 Linux 完全的从我的硬盘中移除呢？！ 呵呵！ 这里就来说一下该如何是好吧！

由于不同的操作系统所使用的档案系统架构(file system)并不相同，有些甚至是不兼容的，例如 Windows 所使用的是 FAT 表，而 Linux 所使用的是 ext2 这个档案格式，这两种格式完全不相同，在 Linux 底下还可以藉由编辑核心来支持 Windows 的 FAT 档案格式，但是 Windows 则完全无法读取 Linux 的档案格式了！此外， Windows 使用的磁盘分割工具 fdisk ，很抱歉的，并不认识 Linux 的 ext2 这个档案格式，所以如果您有一棵已经安装有 Linux 系统的硬盘，呵呵，使用 Windows 的 fdisk 是完全无法分割这块硬盘的！

那么到底什么是硬盘分割呢？真的要将硬盘用刀子割一割吗？!不是这样的，实际上，硬盘是以 sectors(扇区)，cylinder(磁柱)，partitions(分割槽) 这些东西来作为储存的单位，而最底层的实体硬盘单位就是 sectors 了，通常一个 sector 大约是 512 bytes 左右。不过，在磁盘进行格式化的时候，可以将数个 sector 格式化成为一个逻辑扇区(logical block)，通称为 block。blocks 为一个档案系统(filesystem)存取的最小量。 那么 partition 是什么？简单的来说，你知道你的 Windows 有所谓的 C:, D: 是吧！其实他们是同一颗硬盘，只是利用『磁盘分割表』(partition table)来将实体的硬盘规划出不同的区块。

举个例子说，假设你的硬盘总共有 1024 个 cylinder （利用 blocks 结合而成的硬盘计算单位）， 那么你在这块硬盘的文件头地方 （就是磁盘分割表，可以想成要读取一块硬盘时最先读取的地方）如果写入你的 partitions 共有两块，一块是 primary 一块是 extended ，而且 extended 也只规划成一个 logical ，那么你的硬盘就是只有两个槽啦（对于系统来说，真正能使用的有 Primary 与 Logical 的扇

区， Extended 并无法直接使用的！需要再加以规划成为 Logical 才行！），而且在 partition table 也会记录 primary 是由『第 n1 个 cylinder 到第 n2 个 cylinder 』，所以啰，这样子一来，当系统要去读取 primary （就是 c 槽）的时后，就只会在！n1~n2 之间的实体硬盘当中活动啰！

基本上，Windows 98 系统中的 Fdisk 这支程序仅支持一个 primary 与一个 extended ，其中， extended 可以再细分成多个 logical 的硬盘槽。NT 很抱歉，小弟不熟，所以就不提了！那么 Linux 呢？嗯！基本上最多可以有 4 个 primary 的硬盘，而可以支持到 3 个 primary 与一个 extended ，其中， extended 若再细分成 logical 的话，则全部 primary + extended + logical 应该可以支持到 64 个之多。底下我们将以 spfdisk 这个全中文接口的 fdisk 磁盘分割工具来介绍如何分割硬盘！(注：更多详细的磁盘与磁盘分割信息，可以参考 SPFdisk 的官方网站喔！ 在最底下的参考数据当中有提供连结呢！)

硬盘分割 ==> SPFdisk

SPFdisk 是一套由国人开发完成的全中文接口的硬盘分割工具，他要比微软出的 Fdisk 功能强多了， 他的好处有：

- 全中文界面让你一定可以看的懂之外，简单的类图形接口可以让你轻易的进行硬盘分割；
- 除此之外，这套软件的『DOS 工具』内的『格式化工具』格式化硬盘的速度真是 DOS 比不上的，我格式化一个 30GB 的硬盘不用十秒就可以格式化完全！

另外,其它的优点我在此也不多说了,若有需要你可以自行自一些搜寻网站下载最新的程序，或从 这里 下载鸟哥有的程序，不过可能旧一点。

另外，由于 DOS 的 Fdisk 并不认识 Linux 的分割表，所以用 DOS 的 Fdisk 是无法将 Linux 的分割表去除的。因此，你要删除 Linux 的分割表，只有两个比较快的方法，一个是以 Linux 直接再分割，一个则是使用 SPfdisk 分割啦！

Tips:
由于您正在阅读的这个页面的影像档案很大,有时候会有没办法显示的情况发生， 这时请在画面上『按鼠标右键』，再选择『显示图片』这个选项，即可显示画面啦！



硬盘分割主要可分为下面几个步骤：

1. 将旧有的分割表删除；
2. 建立新的主分割及扩充分割（若有需要的话）；
3. 贮存分割表；
4. 以 DOS 工具格式化以分割的硬盘。

---

1. 删除原有的分割：

假设你的主机中没有任何系统存在，则请以 Windows98 制作开机片后，将 spfdisk 拷贝至开机片。以此磁盘开机之后，执行：

```
A:\>spfdisk
```

会出现如下欢迎画面。



按任意键后出现下面画面：



以方向键移动光标至『硬盘分割工具』按 Enter 键后会出现画面如下：



这是向你询问是否需要使用 FAT32 的档案系统，由于 Windows 98 支持的长档名及相关的档案型态是以 FAT32 为准，所以当然按 『Y』！按了 Y 之后会出现硬盘的信息，如下所示：



因为我是在我原有的机器上执行这个程序，所以会有两颗硬盘，如果你的系统只有一颗硬盘的话， 则只会显示你有的硬盘数据，在上图中，1.19GB 的硬盘其总磁柱仅有 621 单位，比可开机扇区范围（0 ~ 1023）小得多，所以可以随意分割。将光标以方向键移动至 1.19GB 这颗硬盘后，按 Enter ：



因为这颗硬盘之前被我灌过 Linux ，所以会显示 Linux 的扇区划分情况。上面的意义为：

- 启动：由于系统开机时会去先去找分割表， 由分割表所设定的『可开机扇区』进行开机程序， 因此若这个扇区为开机扇区，则『启动』项目会有一个心型的符号存在！
- 起始磁柱与结束磁柱：这一个被分割的扇区的开始与结束扇区。
- MBytes：这个扇区的硬盘容量。
- 系统 ID 与系统种类:这一个分割表的类型。因为这是 Linux 的分割类型,所以其 ID 显示为 83，若是 FAT32 的话，则为 0b。

将光标以方向键移动至这个扇区后，按 Enter ：



在这个画面中，将光标以方向键移动至『删除分割』这个项目，并按下 Enter ：



出现此画面后，确定要删除这个分割就按『[Y]是』这个项目。



删除分割之后你的硬盘中就没有分割表的存在了，所以这个硬盘的系统种类则变成了 未规划。

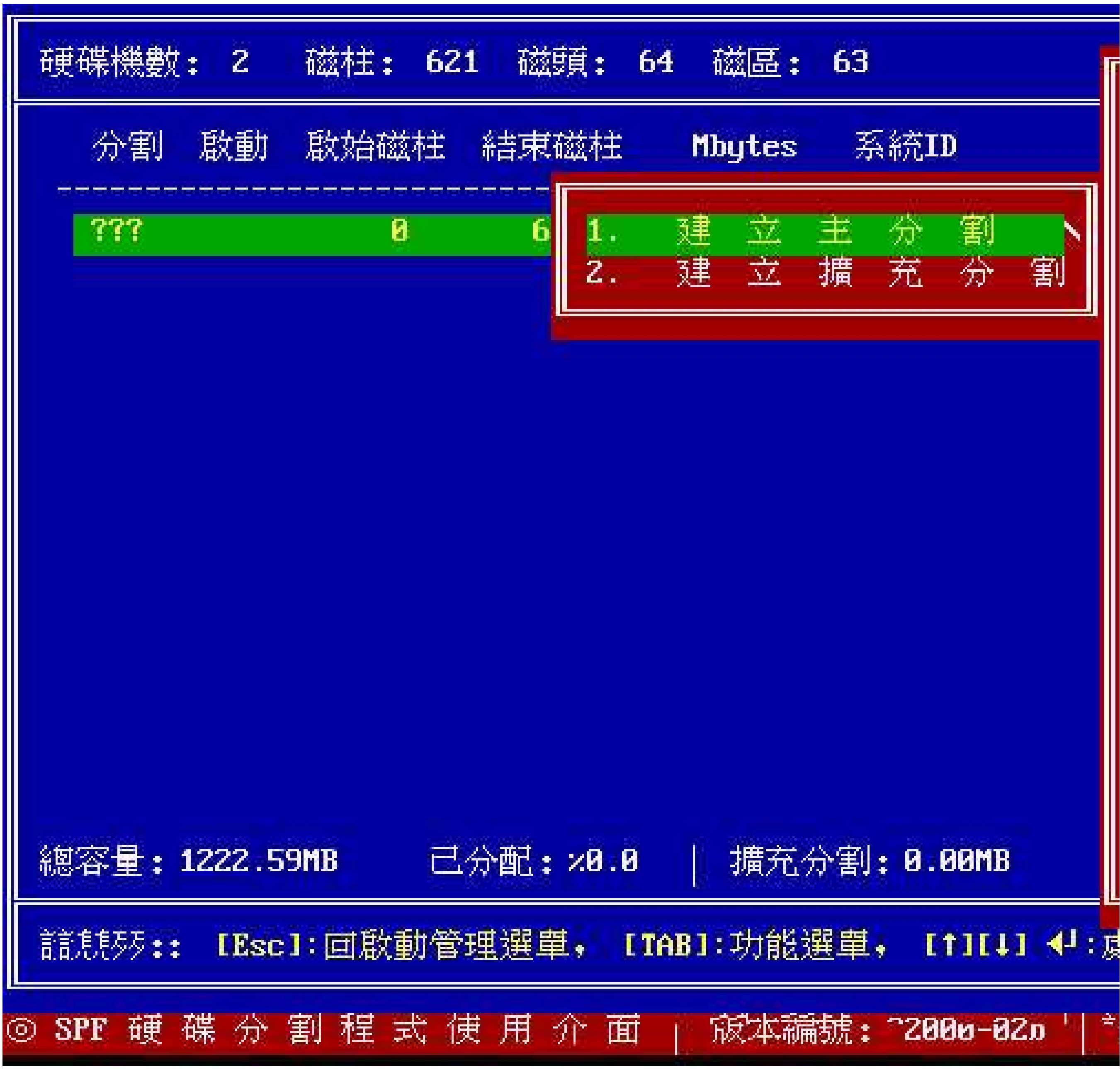
---

2. 建立主要分割扇区：

在上面的画面中，按下 Enter 键，会出现下面画面。



由于这一颗硬盘的分割表被删除了，所以『系统 ID』与『删除分割』被取消了。这时按下『建立分割』会出现如下画面：



然后选择『建立主分割』，那何谓主分割与扩充分割呢？所谓的主分割在 windows 系统下即是『C 槽』啦！但是扩充分割并非『逻辑分割』，这里要注意一下， 所谓的『逻辑分割』是包含在扩充分割中的，例如当你的扩充分割共有 10GB 但是你想将之分为两槽，则可以使用逻辑分割将扩充分割分为两槽， 这两槽即称为『逻辑分割』。所以这里要注意啦，建立扩充分割的时候就要选择『配置整个区域』啦！ 好！下一步按下『建立主分割』：



这里会问你是否要将整个硬盘分割为仅有一个磁盘区？由于我们要将硬盘分为两槽， 所以这里当然选择『[N]否』啦！



由于你选择了『不要配置整个区域为一块扇区』，所以这时程序要你输入你所需要的扇区。 通常在第一步是输入『启始磁柱』，这时只要按 Enter 就可以啦，然后会要你输入『结束磁柱』， 结束磁柱的输入方法有两种模式，一种是输入磁柱区，一种是输入你所需要的 MB 数，通常我是输入 MB 数啦，例如如上所示，我所需要的空间大小是 600MB ， 所以输入『+600』即可，而如果你的硬盘很大，你要输入 4GB 时，则需要输入 『+4000』，以此类推！输入『+600』并按 Enter 之后出现如下画面。



这时出现了你刚刚划分的硬盘信息啦，由于我们是划分为 DOS 分割区，所以系统种类是 FAT-32 ，而 ID 则为 0b 。至于另外尚未划分的就会显示为〈未规划〉啦！如果你还需要再继续划分的话，这时将光标移动至〈未规划〉的那一个扇区按 Enter 后选择『扩充分割』即可继续划分。如果划分完毕之后， 当然就是贮存分割表啰。这里注意一下，因为刚刚的动作均尚未完成贮存的工作，所以要反悔还来得及！

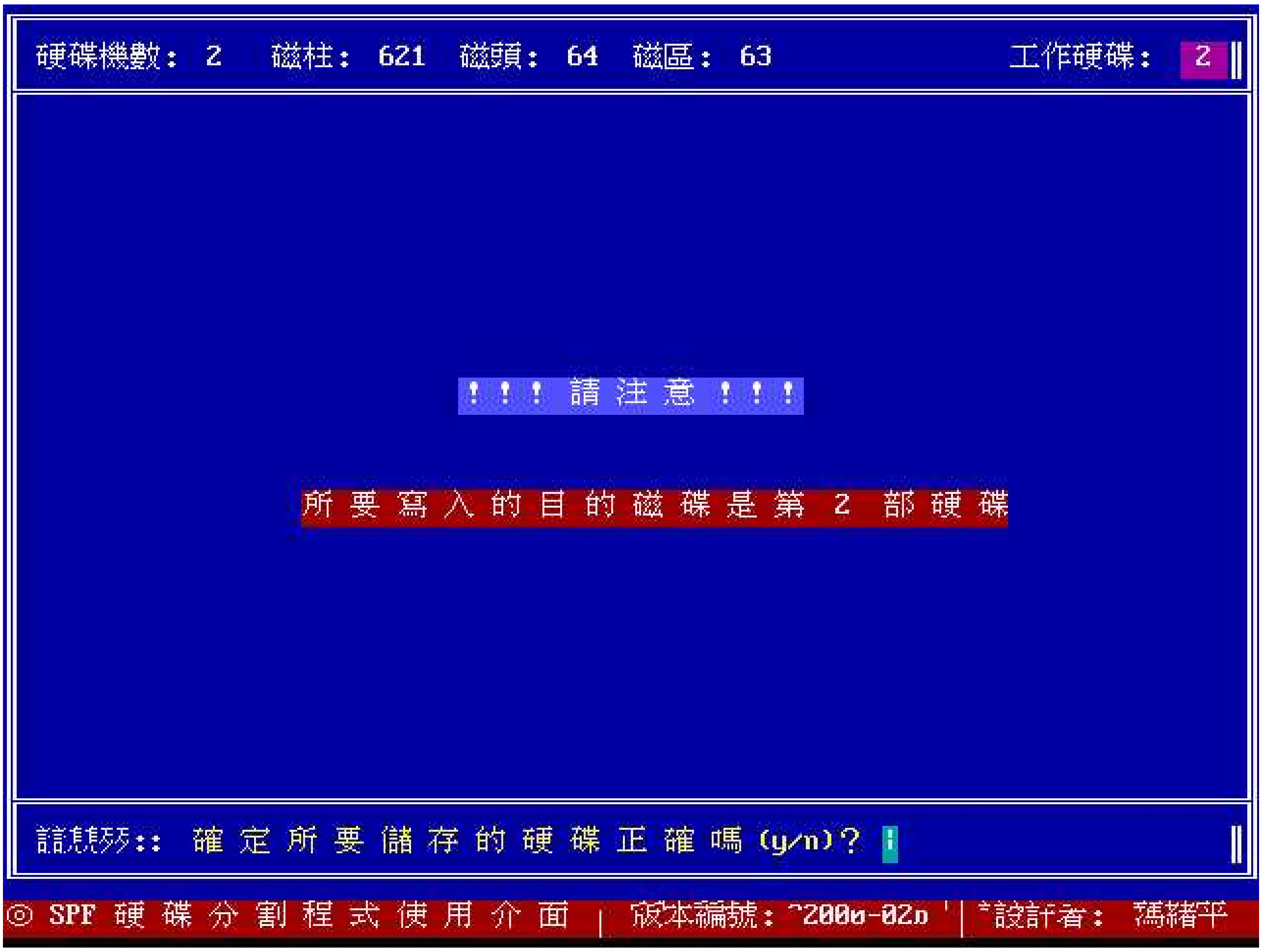
---

3. 贮存分割表：

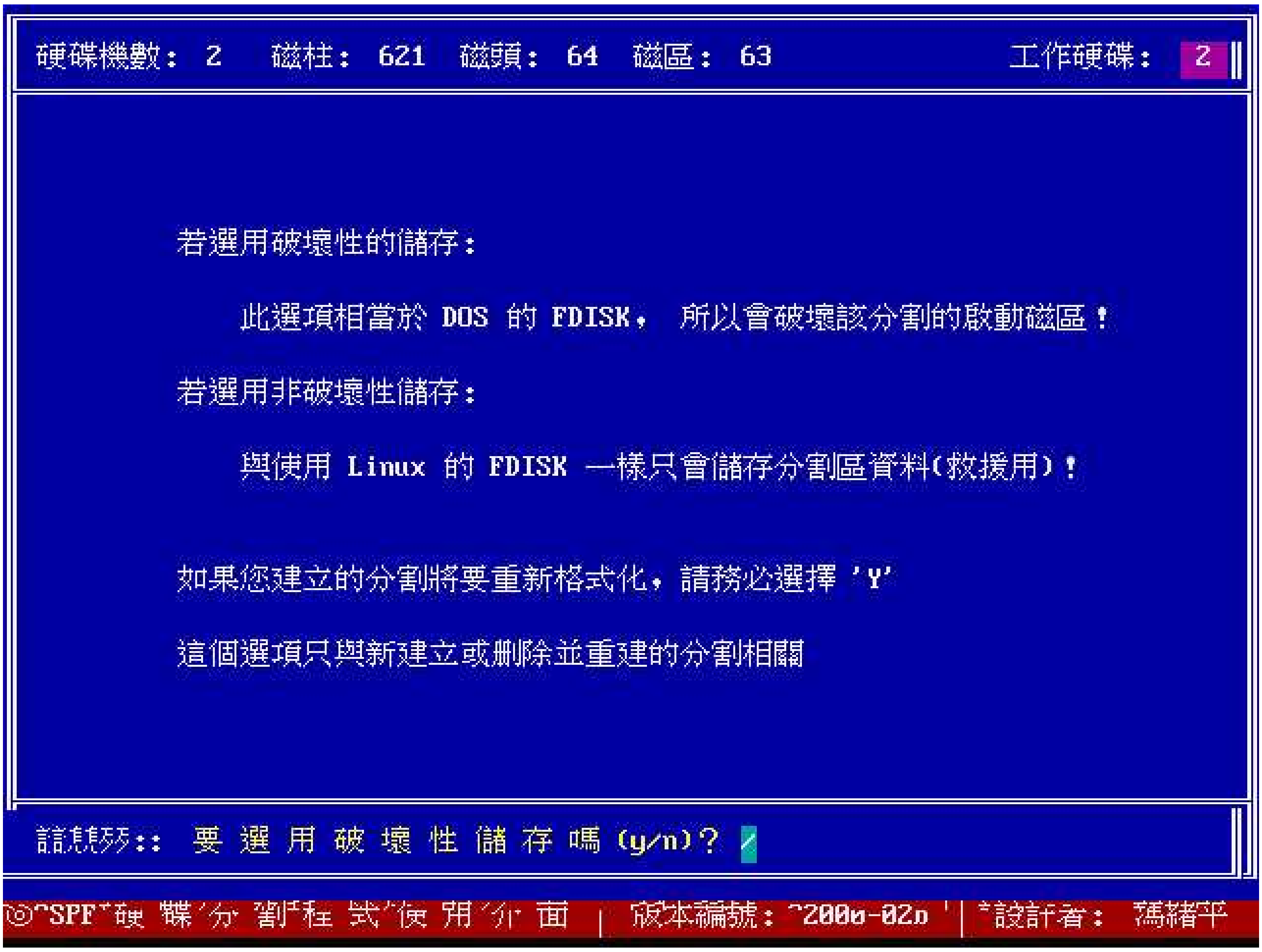
按下来要做贮存的动作了，按下『Esc』键 （键盘左上角那个键）后会出现如下画面：



按『[Y]是』，将刚分割好的分割表贮存至硬盘中！ 然后出现如下画面：

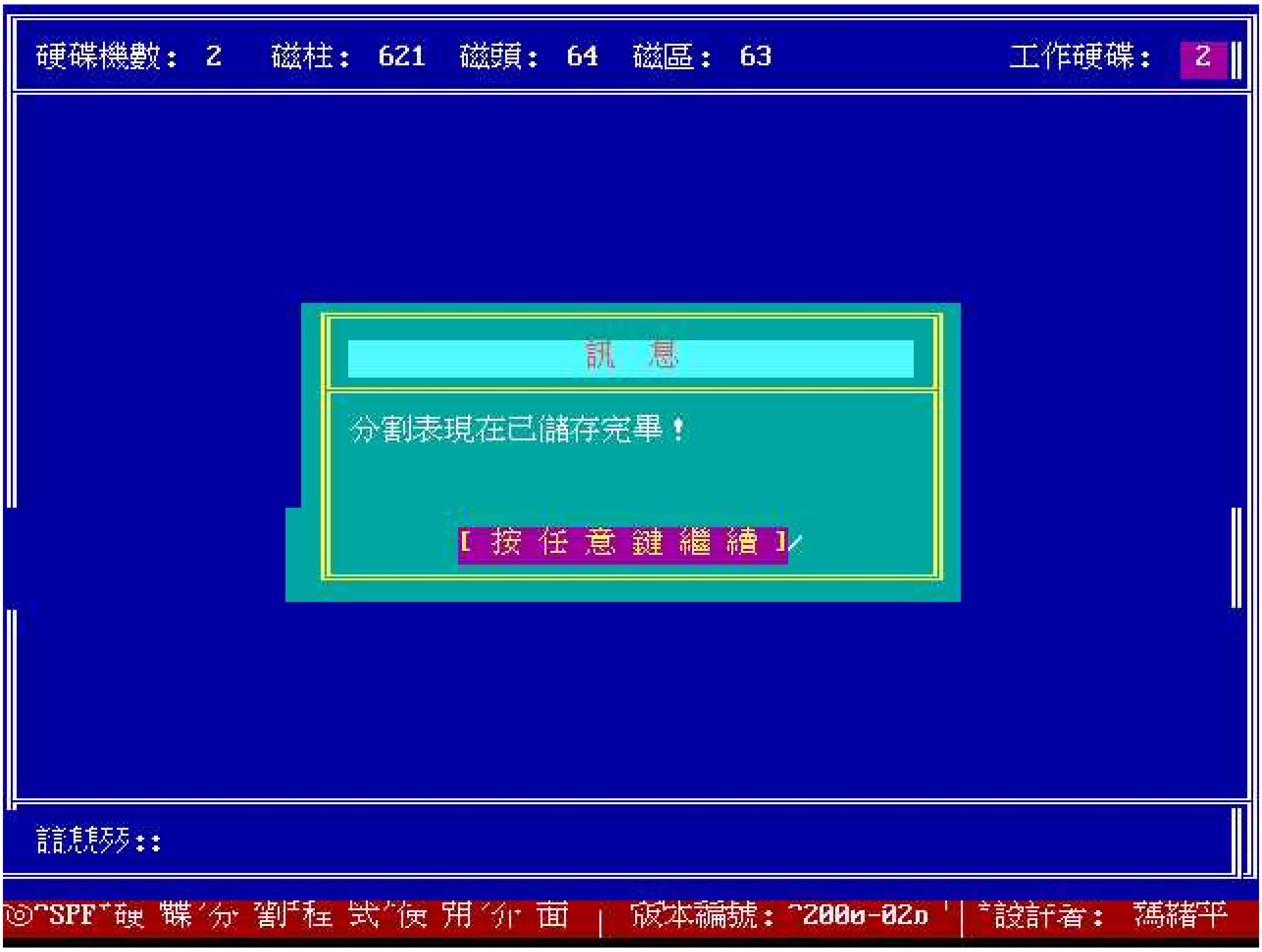


硬碟機數: 2 磁柱: 621 磁頭: 64 磁區: 63 工作硬碟: 2
!!! 請 注 意 !!!
所 要 寫 入 的 目 的 磁 碟 是 第 2 部 硬 碟
確 定 所 要 儲 存 的 硬 碟 正 確 嗎 (y/n)?
◎ SPF 硬 碟 分 割 程 式 使 用 介 面


硬碟機數： 2 磁柱： 621 磁頭： 64 磁區： 63 工作硬碟： 2
若選用破壞性的儲存：
此選項相當於 DOS 的 FDISK， 所以會破壞該分割的啟動磁區！
若選用非破壞性儲存：
與使用 Linux 的 FDISK 一樣只會儲存分割區資料(救援用)！
如果您建立的分割將要重新格式化，請務必選擇 'Y'
這個選項只與新建立或刪除並重建的分割相關
要 選 用 破 壞 性 儲 存 嗎 (y/n)?



硬碟機數: 2
磁柱: 621
磁頭: 64
磁區: 63
工作硬碟: 2
交 談
是 否 建 立 UNDO 檔 ?
[Y]是 [N]否


硬碟機數： 2 磁柱： 621 磁頭： 64 磁區： 63
工作硬碟： 2
訊 息
分割表現在已儲存完畢！
[按 任 意 鍵 繼 續]
版本編號：
設計者：



这里的动作是连续的：

1. 程序会先跟你确认你的硬盘有没有错误，这里还可以反悔。
2. 然后程序会问你是否需要使用破坏贮存，一般来说是需要使用『破坏贮存』的， 因为需要将你的硬盘划分完全啦！所以要按 [Y]；
3. 为了可以让你以后回复分割情况，所以你可以选择 『建立 UNDO』档，所谓的 UNDO 文件即是记录你之前硬盘分割表信息的档案啦。

这样一来你的硬盘就划分完毕而且贮存啦！这时要做的就是重新开机并格式化硬盘。格式化硬盘可以使用 DOS 的 Format ，当然也可以使用 spfdisk 的内建功能喔！

---

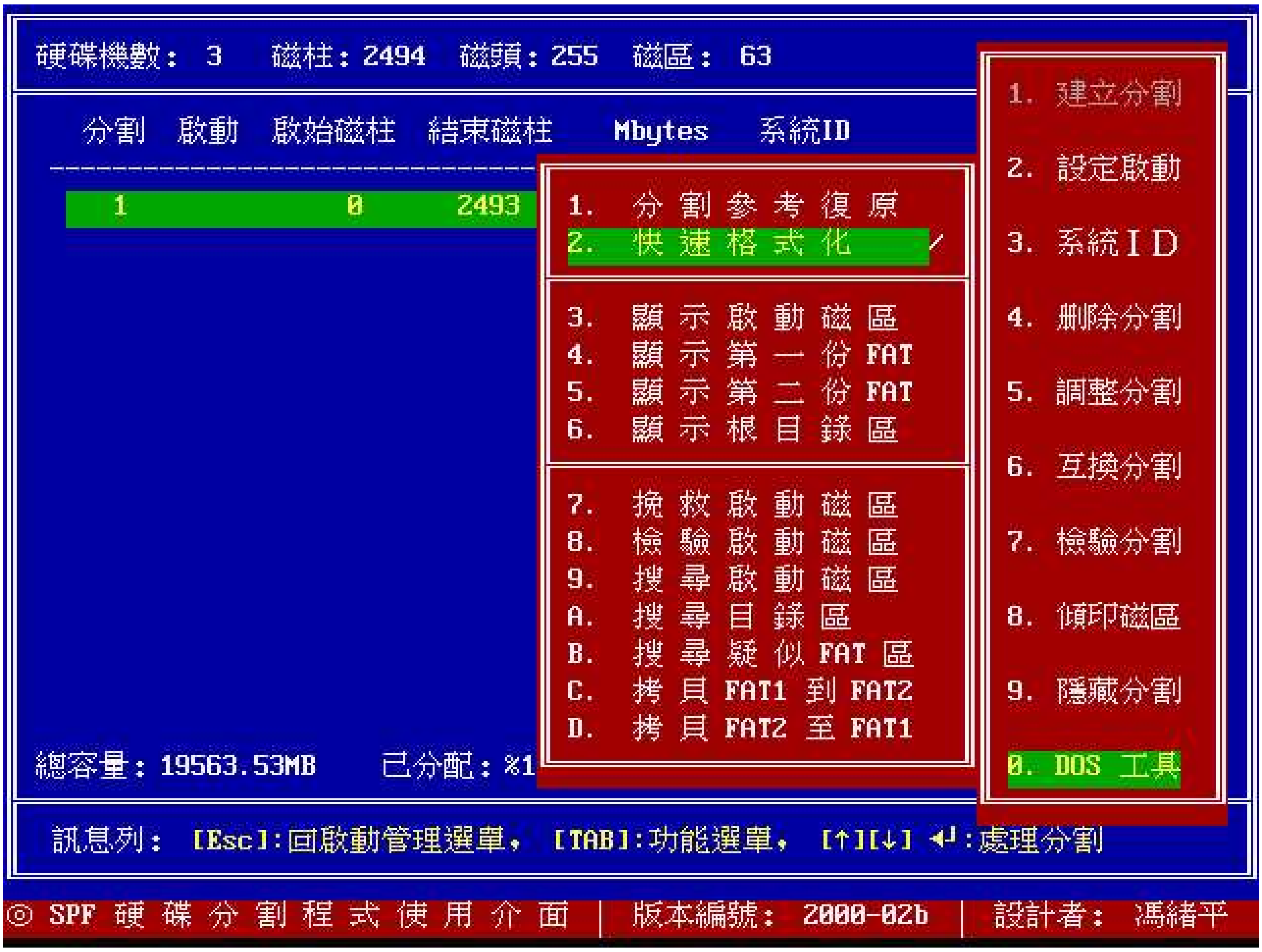
4. 格式化硬盘：

再进入刚刚你划分完毕的那个硬盘区，按下 Enter 之后会出现一串选单，然后最下方的选单为 『DOS 工具』，选择这一项并按 Enter 后，会出现另一个选单：



在这个次选单中的第二项即是快速格式化，这个格式化的动作非常的快喔！比 DOS 的格式化快多了！不过，

这里也必须指出一个问题，那就是若你的硬盘有坏轨的话，那最好还是使用 DOS 的 format 比较完整一点。



参考数据

- SPFdisk http://spfdisk.sourceforge.net/
  事实上， SFPdisk 能作的事情还很多，包括最为人所熟知的 boot loader 的应用！ 建议您一定要到 SPFdisk 的官方网站上面瞧一瞧喔！ ^_^

简易且较小安装 Red Hat 7.2

最近更新日期：2003/02/03

主机的硬件配备与预计开放的服务

就如同前面所说的，安装你的 Linux 之前，最好先来了解一下您的 Linux 用途！当然啰！如果目前您所需要的 Linux 仅只是在于学习 Linux 的指令的话，那么底下的咚咚你都可以把他看看就好！不过，还是强烈的建议您慢慢的一步一步的安装你的 Linux 系统，这样对于您的 Linux 系统会有比较完整的概念说！此外，由于 Linux 系统最好在安装完成之后，立即重新编译过他的核心，以使 Linux 系统『较为稳定』，所以，在你可以连上 Internet 的时候，千万记得一起下载新的 核心 喔！由于 Red Hat 7.2 的核心版本为 2.4.7 ，因此，你必须要下载较新的 2.4.17 以后的版本喔（到 2002/02/16 为止）。

我的主机配备：

好了，既然 VBird 写的这个部分主要的目的是在于『使用淘汰的计算机来进行 Linux 服务器的安装』，那么我的硬件配备当然不会太好啰！以下就是我的配备啦：

- CPU 为 P-166，主机板为华硕的老主机板；
- 使用 64 MB 的 RAM （是 72 pin 的喔！）；
- 硬盘为 3.2 GB 的硬盘，安插在 primary 排线的 master 上面；
  这里要特别说明一下，通常在 586 之后的主机板上面都有两条接排线的界面（排线就是硬盘与主机板相接的那一个东西啦！），而我们称这种界面为 IDE 界面（目前的主流硬盘界面），并且主机板上面的这两个界面就分别称为 Primary （主要的） 与 Secondary （次要的） IDE 啰。
  而如果你有仔细观察的话，那么每一条排线上面还有两个插孔，也就是说一条排线可以接两个 IDE 界面的装置（硬盘或光驱），而你有两条排线，因此一个主机板在预设的情况中，应该都可以接四个 IDE 界面的装置。好了，那么每条排线上面该如何判别哪一个是主硬盘（ Master ），哪一个是副硬盘（ Slave ）呢？基本上这个需要调整硬盘上面的 jump 才可以知道！这个时候，请察看一下您的硬盘机吧！上面应该都会有图示说明才对！
  这一部份请特别留意喔！因为不同的硬盘接法将会导致不一样的状况，更严重的，将会导致无法开机的窘境，所以建议您注意一下这里！
- 网络卡预计使用两块螃蟹卡，不过，如果您不希望有无法分辨网络卡的状况发生，那么建议使用两块不一样芯片的网络卡比较好！
- 显示卡使用的是 S3 Virge 的 PCI 显示卡，不过由于我以后的过程中将不会使用 X-Windows ，所以这部份似乎不会有问题！
- 安装过程中需要的装置：键盘、屏幕、光驱、软盘机等等，这些装置在安装完成 Linux 之后，即可马上拔掉！

硬盘 partition 的问题：

硬盘的 partition 是相当重要的一环呦！这里有一些重要的信息要先跟大家报告！就是说：

- 在 Linux 底下，每一个装置都以一个档案来代表，例如 IDE1 的 master 为 /dev/hda ，而由于 primary + extended 最多有四个 partition ，所以第一个由 extended 分割出来的 logical 扇区为 /dev/hda5 ！
- 需要特别留意的另几个装置是网络卡、软盘、光盘，其代号分别为： eth0, /dev/fd0, /dev/cdrom!
- 如果你有一个硬盘接在 IDE2 的 master 上面，并且有 5 个可以使用的扇区，同时你分割了 2 个 primary partition 时，那么你的磁盘应该就会有底下几个代号：
    - /dev/hdc1 (primary)
    - /dev/hdc2 (primary)
    - /dev/hdc3 (extended, 这个为不可使用的磁盘代号)
    - /dev/hdc5 (1st logical)
    - /dev/hdc6 (2nd logical)
    - /dev/hdc7 (3th logical)

预计开放的服务：
虽然是老旧的配备，不过相对于我服务的机器数：五部个人计算机，也相当足够了！呵呵！那么我需要的服务有哪些呢？

- NAT ： 用来分享频宽；
- Mail ： 用来收发信件；
- WWW ： 用来给大家架设个人网页；
- Proxy ： 用来提供五部区域计算机内的用途，并加以分流；
- DHCP ： 主要在提供内部计算机不需要安装一些有的没的！
- FTP ： 最好是不要安装的啦！

我的网络：
我的网络主要是以 拨接制 ADSL 为主，那么如何规划呢？

- 在 Linux 系统中，预计以 rp-pppoe 这个软件来拨接 ADSL 并且予以分享；
- 我的内部网络之 网段 为 192.168.1.0/255.255.255.0 这一个，没有再切割的子网络；
- 我的 Linux 主机名称为 vbird.adsldns.org ，是跟 www.adsldns.org 申请的动态 DNS 系统；

选择的套件：
我选择的安装套件为 Red Hat 7.2 版，他的特征为：

- 这一版的预设核心为 2.4.7-10 ！
- 预设的防火墙机制为 Kernel 2.4.x 的 iptables ；
- 预计使用 LILO 作为 Boot Loader 喔！
- 另外，由于在设定其它的服务之前，想要先以较新的 Kernel（ 核心 ）来编译过，因此需要先下载核心！

大致上就是这样啰！

硬盘规划

自订安装『Custom』：
初次接触 Linux ：只要切割『 / 』及『 Swap 』即可!
好了，通常初次安装 Linux 系统的朋友们，我们都会建议他直接以一个最大的扇区『 / 』来安装，这样有个好处，就是不怕分割错误造成无法安装的困境！例如 /usr/ 是 Linux 安装程序中摆放的目录，万一你分割了一块扇区给 /usr ，但是却给的不够大，那么就伤脑筋了！因为会造成无法将数据完全写入的问题，就有可能会无法安装啦！因此上，如果你是初次安装的话，那么可以仅分割成两个扇区『 / 与 Swap 』即可！
建议分割的方法：预留一个备份的扇区！
就如同前面几个心得分享文章中提到的，由于 Linux 预设的目录是固定的，所以：

- 通常我们会将 /var 及 /home 这两个目录稍微加大一些，如果硬盘够大的话，加个几 GB 也不为过！
- 另外， /usr 至少给他 3~5 GB 吧，如果硬盘真的大的话！
- 而 / 也可以给个几 GB 的空间。
- 最后，由于我们的 Linux 可能是在『试用』阶段，所以很有可能会重复的一再安装，因此上，我都会预留一个扇区来备份我的核心啦与实验过程中觉得不错的 scripts （就有点像 DOS 的批次檔），当然，我的 /home 底下的咚咚也可以有备份的地方，而安装套件的源文件也可以摆在这里！有个最大的好处是，当我的 Linux 重新安装的时候，我的一些套件马上就可以直接在硬盘当中找到！呵呵！重新安装比较便利啦！

选择 Server 的硬盘切割方式：
对于首次接触 Linux 的朋友们，通常不建议使用 Red Hat 预设的 Server 安装方式，因为会让你无法得知 Linux 在搞什么鬼，而且也不见得可以符合你的需求！不过，这里仍然说一下选择 Server 的时候，他是如何切割硬盘的呢？

注意：选择 Server 的时候，请『确定』您的硬盘数据是不要的！因为 Linux 会自动的把你的硬盘里面旧有的数据全部杀掉！此外，硬盘至少需要 2 GB 以上才可以选择这一个模式！

- 64 MB 的 Swap ；
- 256 MB 的 / ；
- 256 MB 的 /var ；
- 其它的空间平分给 /usr 与 /home ！

知道了吗？由于 Server 会有上面的限制，所以通常我都不太喜欢让 Linux 自己切啦！选择 Custom 比较好说！
硬盘的代号意义？
在 Windows 或者是 DOS 年代，硬盘以 FAT 表来切分时，他们的代表扇区为 C:D:E:...但是在 Linux 中则不然喔！一个『目录』可以代表一个『装置』！基本上，每一个硬盘在安插的 IDE 接口中，都有不同的代号：

硬盘安插的 IDE 接口	Linux 上面的磁盘名称
第一个 IDE 的 Master 上之硬盘	hda
第一个 IDE 的 Slave 硬盘	hdb

第二个 IDE 的 Master 硬盘	hdc
第二个 IDE 的 Slave 硬盘	hdd

另外，需要特别留意的是，每一个硬盘（例如 hda ）最多可以有 4 个 primary 扇区！分别是 hda1, hda2, hda3, hda4！而如果是逻辑扇区的话，那么就需要由 hda5 开始增加啰！

多重操作系统的安装流程

- 硬盘重新规划的多重开机系统：

  如果你想要在你的 Linux 机器上同时安装 Windows ？可行吗？当然可行啰！况且目前很多的朋友手边只有一部计算机，但是又想要同时学习一下 Linux ，呵呵！那么安装多重操作系统实在是必须要的！好了！那要如何安装呢？以我前一阵子帮一个朋友规划的 Win98, Win2000, Linux 为例，我先将硬盘以 spfdisk 切割成两个 FAT partition，分别是 2GB 与 3GB ，预计安装 Win98 与 Win2000 （分别是 C: 与 D: ），然后再以 CD 开机后，分割最后的磁盘成为 / 与 Swap 两个！好了！如何安装：

  1. 先以 Spfdisk 分割硬盘：由于 Windows 的 Fdisk 实在太慢了，我蛮喜欢使用 spfdisk 这个全中文的磁盘分割接口的！简单又方便！将硬盘切割成 C: 2GB, D: 3GB即可！详细的 Spfdisk 执行范例可以看一下底下这一篇： spfdisk 范例
  2. 先安装 Win98 ：这个简单吧！用 98 开机片开机之后，直接安装，并且选择安装在 C 槽即可！
  3. 再安装 Win2000：进入 Win98 之后，将 Win2000 的光盘片放进光驱中，屏幕会自动的跑出一个窗口，问你要不要升级，选择『是』，然后会进行一些小动作！在安装程序问到『升级安装或全新安装』的时候，请千万选择『全新安装』这个项目，并且不要升级硬盘扇区！然后在出现一个『问你安装目录所在』的问题时，进入选项里面，选择『要我自己挑选硬盘分割区』那个项目！然后接下来一直按下『确定』或『是』即可！之后，计算机会重新开机，开机完成之后会进入 Win2000 的安装画面，然后在出现『安装扇区』的时候，请选择 D 槽，并且选择『不要更改扇区档案系统』即可！接下来就会完成一些程序啦！
  4. 最后才安装 Red Hat 7.2：是的，最后才安装 Linux ！安装的过程底下会说明喔！
  5. 以 Lilo 设定多重开机：是的，我还是比较习惯使用 Lilo 来作为多重开机的设定啦！

  好了！这样你就可以具有多重开机的主机系统啰！很高兴吧！呵呵！先别高兴的太早！很多的朋友安装 Windows XP 及 Windows ME 版本与 Linux 共存，安装的结果是『残念』的！不过个人没有试过 XP 与 ME ，所以无法提供任何的解答！这点请千万注意了！在我的经验中，使用 98 与 2000 来与 Linux 共存是没有问题的（在我的旧机器与新的双 CPU 主机当中都试过！ OK ！）

- 在既存的 Windows 系统中加装 Linux 系统：

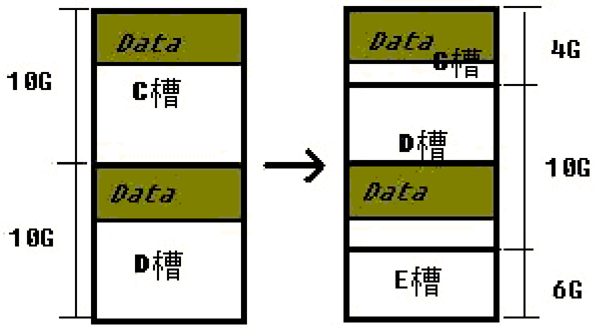
  另外再提供一个之前也曾经安装过的一个经验！恩！你可能会觉得奇怪，这个方法跟上一个方法有什么不同！？呵呵呵呵！最大的不同在于：

- 我既存的 Windows 系统中的数据不想丢掉，并且我也没有新的硬盘来暂存我的系统或者是备份数据！假设原本我的 20 GB 硬盘中分割成 10GB, 10GB 两槽，但是我还想要安装 Linux ，且是在『旧系统仍然可以存活』的情况下！那该如何是好？！

这真的是很有趣的问题！早先在 Windows 系统中， VBird 就犯了一个错！C 槽给的太大了！基本上，系统文件不需要太大啦！通常我都喜欢 C 槽只给大约 4 GB 左右的空间（甚至更小），这是因为 C 槽是很需要备份的！如果太大的话，备份很麻烦！所以系统重置就会很花时间（因为所有的东西都要重新安装！我哩咧....！）！因此，我都习惯将 C 槽只给一点点的空间，然后再安装完并设定完所有的系统之后，马上以 Ghost 来备份我的系统！而所有的备份数据文件都摆放在 D 槽！此外，我的 OutLook Express 的书信目录也都不是摆在 C 槽！呵呵所以我不会很害怕 C 槽挂掉，因为，直接以 Ghost 还原即可啰！系统还原还不需要 30 分钟呢！

这里就发生一个问题啦，假如原本的系统是 10GB, 10GB 的两槽，不过全部的有用到的资料量只有 10GB 不到！也就是还有空间来安装 Linux ，但是由于硬盘切割的不好，所以伤脑筋！此外，我的原系统希望留下来，而且也希望可以安装 Linux ，要怎办？！我曾经这样做过：

- 由于 FAT 的扇区使用，其实只是在磁头区域（所谓的硬盘第零轨）规划而已，所以，我就将我的数据先以『磁盘重组』的方式将数据都归在一起；
- 然后以 Spfdisk 将该硬盘的 FAT 表进行分割，注意喔！只是分割 FAT 表，并没有 format 喔！不过这里的技术性很高，需要特别注意！因为你是将 FAT 表重新划分，所以你的数据必须要在同一个扇区内！好了，我就将原本的 10GB 10GB 切割成 4GB、10GB 与 6GB 三槽！而且在 spfdisk 的帮助之下，顺利的在没有任何数据遗失的状况下，将我的硬盘由原先的两槽分割成三槽啰！那么一来，我就可以在我原本的 D 槽里面安装 Linux 啦！方法有点像底下的图示：



很神奇吧！数据还是在原来的地方，不过扇区的定位点改变了，还多出一个扇区！不过，这里要提醒大家，虽然 VBird 曾经以这个方法成功而且完全没有惊险的将硬盘数据在不毁损的情况下，顺利的将硬盘切割完毕！但是那是小弟已经了解到 FAT 与扇区的相关性（其实 FAT 只是在『规范』你的硬盘读取头读取的『头』跟『尾』而已，并不是真的将硬盘『切割』啰！），而且我也有『壮士断腕』的觉悟！呵呵！因此不是很建议您这样做！尤其是当你的数据还很重要的时候！切记切记！

Linux 安装流程（较小安装、完全安装）

接着下来要开始来安装 Linux 啰！说了这么多有的没的 ㄟ.... 不好意思， VBird 太喜欢胡扯了....

事先检查：

基本上你必须先检查一下你的武装配备喔：

- 下载并烧录 Red Hat 7.2 的可开机光盘：不要问我如何烧录～～

  enigma-i386-disc1.iso
  enigma-i386-disc2.iso

  强烈的建议您不要使用 HTTP 来捉这两个档案，就是不要使用 IE 或者是 Netscape 之类的浏览器来捉这两个档案，因为档案太大了，在传输的过程中会有捉错的情况，所以就会很麻烦....建议使用续传软件，或者是直接以 FTP 软件到中山大学的 FTP 站捉，不但具有续传功能不怕断线，捉到的档案也会比较正常（已经有很多朋友在 BBS 上面留言，发现捉的档案无法烧录啰！）。此外，也建议直接下载新的核心，以方便后来的核心编译工作！目前新的核心已经出到了 linux-2.4.17 （截至 2002/02/16 为止），通常 VBird 都是在中山大学下载的，你可以到底下来看看呦！

  中山大学关于核心

- 进入 BIOS 设定开机顺序：
  基本上如果是不太旧的主机板都会支持光盘开机的，使用 CD-ROM 开机的好处是比较快，而且也不用再去做 Linux 安装开机片，确认的方法如下：
  - 按电源键开机；
  - 在进入系统之前会出现 Del 字样（每个厂牌不太相同），此时按下键盘上的 Delete 键；
  - 进入 BIOS 之后以方向键选择 『BIOS Features Setup』这一项，或者是『Advanced BIOS Features』，不管如何，反正只要看到 『BIOS Features』字样的那一项就对了！；
  - 将方向键移动至 『Boot Sequence』 或者是 『First Boot Device』； 这一项，按键盘上的『Page Up』 或 『Page Down』 按键，选择『CD-ROM』 为第一开机顺位即可。这里注意一下，如果你的机器并不支持 CD-ROM 开机的话，你一定找不到 CD-ROM 这一项，这时请制作开机片吧，并将此项调整为『A』为第一顺位；
  - 按键盘上 『ESC』 键退出；
  - 将方向键移动至 『Save and Exit』 这一项按 『Enter』 及 『Y』 确认后重新开机即可！

- 制作 Linux 开机片：
  a. 随便找一台 Windows 计算机，开启 MS-DOS 窗口；
  b. 将可开机 Linux 光盘放入光驱中，在 MS-DOS 窗口键入：

  ```
  C:\WINDOWS> cd   E:\dosutils
  ```
  上面的 E 为你的光驱代号；

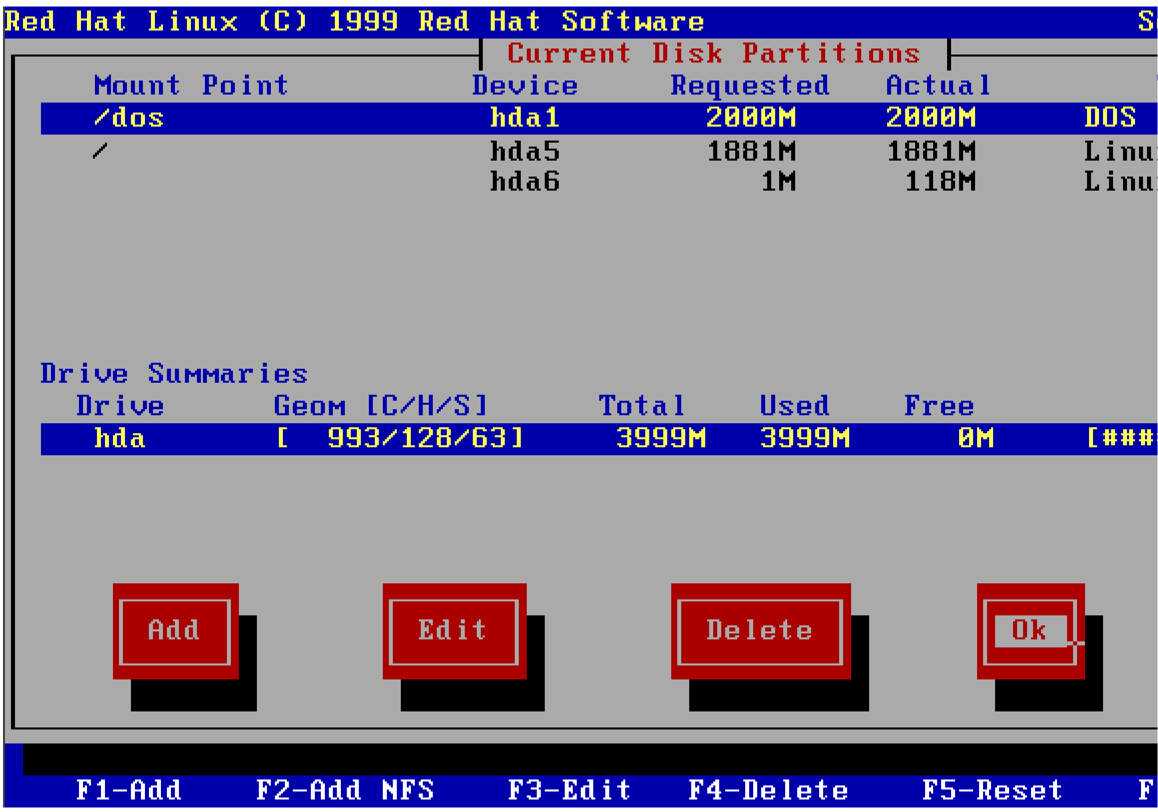
  c. 在 DOS 提示字符下键入：

  ```
  E:\dosutils> rawrite   -f   e:\images\boot.img   -d   a:
  ```
  上面的 e 为你的光驱代号，这时在软盘机放入一片空白的软盘片后，按『Enter』即可。

开始安装：

这样就准备妥当了！正式进入安装吧！特别说明，由于 VBird 不太喜欢使用 X-Windows 系统，所以通常我都使用文字接口安装的，因此底下将以 文字 接口作为介绍，而且，由于许多画面不是很重要，因此 VBird 并没有将画面秀出来喔！

1. 开机==>放入 Red Hat 7.2 的光驱后，以 CD-ROM 开机或者以刚刚做好的 Linux 软盘开机；
2. 选择安装模式==>进入欢迎画面，之后在『 boot: 』的地方输入：『 text 』以文字接口安装！这个时候 Red Hat 会加载一些模块，所以会花费一些时间。
3. 选择语系==>然后在选择语系的地方输入『 English 』；因为文字接口好像没有支持中文的样子！？
4. 键盘模式==>同样的，键盘先选择『 us 』即可；
5. 鼠标模式==>由于我没有鼠标，所以直接按『 tab 』键到『 OK 』按下 Enter 即可；
6. 欢迎画面==>按 Enter 即可；
7. 选择系统模式==>如前所述，这里共分为 Workstation, Server 与 Custom 等等，由于 Workstation 与 Server 会将你旧有的硬盘 Partition 给杀掉，因此我们就直接以『 Custom System 』来安装吧！
8. 要不要 Linux 自动帮你规划硬盘==>开什么玩笑！当然要自己规划自己的硬盘啰！请选择『 Manually partition 』这一项；
9. 选择硬盘分割工具==>硬盘分割工具当然是选择比较简单的啦！那么我们就选择『 Disk Druid 』这个有点像图形接口的咚咚吧！
10. 硬盘分割==>进入 Disk Druid 接口之后，应该有点像底下的图，不过由于 VBird 不会捉图，所以底下的画面是『错误的』喔！ Red Hat 7.2 版已经不是这个样子的图示了！因为他还有加入 ext3 呢！呵呵！不过基本的使用方式还是差不多啦！底下来说一说吧：



基本上你会看到类似上面的话面，总共会显示你的目前硬盘的扇区，如上面说的，『 Primary IDE 的硬盘中的 Mater 为 hda 』！这点请特别留意啰！好了，如果你要将旧有的 FAT 扇区安装 Linux 的话，那么你就必须将该扇区『 Delete 』掉才行，不论如何，请看一下你的剩余硬盘数据空间（注：常常有很多的朋友来信问到，咦！我的 E 槽明明还有 5GB 的空间，为什么不能安装 Linux 呢？！就是这个问题啦！因为 Linux 的扇区与 Windows 的 FAT 并不相同呀！所以你必须将原有的 FAT 扇区砍掉后，才能规划出新的 Linux 扇区呀！所以要看一下硬盘的 Free 喔！）我这里建议的分割方式有几种（请注意，第三个安装的选项是关于 Proxy 的设定方面，如果你要安装 Proxy 套件的话，才建议多加这些扇区！因为据说这样分割的硬盘会让 Proxy 的效能比较好！）：

较小安装（或初次安装）	建议安装	含有 Proxy
o Swap 约 100 MB ； o 其它的都给 /	• Swap 约 100 MB； • /var 给 3～5 GB； • /usr 给 3～5 GB； • / 给 1 GB 以上； • /home 可以给大一些； • /backup 用来做为备份的扇区	o 与 建议安装相同 o /proxy1 给 500 MB； o /proxy2 给 500 MB； o /proxy3 给 500 MB； o /proxy4 给 500 MB

另外，进入每一个扇区之后，你必须要决定：
『 Mount point 』就是扇区啦；
『 Filesystem type 』除了一定要有一个 Swap 之外，你可以选择 Ext3 这个新的扇区喔！似乎有稍微快一点呢；
『 hda, hdb 』这个是硬盘啦！这里请小心选择！
『 Fixed Size 』由于我们都需要给每一个扇区固定的大小，所以这里就选择 Fixed Size 这一个，但是在最后一个扇区（通常是 /backup 这一个）时，我通常都会选择『 Fillall available space 』将其它剩下的空间都给他！
『 Force to be a primary partition 』除非特别需求，例如你的这个扇区是开机区，但是却可能落在 8 GB 以后的扇区内，那么才需要将这个勾选，否则这个选项不要管他！
『 Check for bad blocks 』除非你的硬盘是有坏轨的，否则『千万不要选』不然硬盘检查真的好慢....好慢.....
VBird 的分割结果（3.6 GB 硬盘）：

/dev/hda1 2204 /
/dev/hda2 996 /var
/dev/hda3 502 /backup
/dev/hda5 100 /proxy1
/dev/hda6 100 /proxy2

/dev/hda7 100 /proxy3

/dev/hda8 100 /proxy4

/dev/hda9   64 Swap

11. 选择安装的开机管理系统==>Red Hat 7.2 提供两个开机管理系统，由于 VBird 比较习惯使用 Lilo ，所以这里我是选择『 Lilo Boot Loader 』的！如果你要试一试其它的多重开机控制软件，不反对啦！但是，这里 VBird 仍是以 Lilo 来作为说明的！
12. 选择开机管理系统安装的扇区==>如果没有特殊的需要，就直接选择『 MBR Master Boot Record 』吧！
13. 加载额外的模块==>这个选项专门提供给系统中特殊装置使用的！由于我们的装置都很普通，所以这里就按『 OK 』跳过去吧！
14. 开机系统的名称==>你可以选择其它的名称，当然也可以不用理他，以预设的名称输入之，例如，如果你已经存在有 Windows 系统，那么很可能 Lilo 秀出来的却是 DOS 字样，你可以修改啦！这个可以在未来 Lilo 的部分说明，所以先不要管啦！
15. 网络卡设定==>『先再次强调，VBird 这一次的安装是预计要安装两块螃蟹卡的，由于具有相同的芯片组，所以我在安装的时候仅先安装一块而已！并且，这一块网络卡预设是做为内部虚拟网络之用的！另外一块会在后面才安装上去！』。如果你不是使用很奇怪的卡，那么这个步骤应该会出现网络卡的设定的！（我在这个步骤中预设是当作内部网络之用！所以先给他一个虚拟 IP 喔！）

    首先将 DHCP 前面的 [*] 取消（按空格键）；
    然后按上下键来设定你的网络条件成为：

    IP：192.168.1.2

    Netmask：255.255.255.0

    Gateway：192.168.1.2

    Primary DNS：139.175.10.20

    Secondary DNS：163.28.112.1

16. 主机名称的设定==>给自己一个名字吧！例如我的主机为 vbird.adsldns.org 啰！这里写错也不要紧，后面会提到修改 host name 的方法！
17. 防火墙的设定==>由于我们会在后续的步骤中更改一些防火墙机制，所以这里不用设定啰！选择『 No Firewall 』那一个项目；
18. 选择语系==>通常我只选择两个语系，分别是『 English 』与『 Taiwan, R. O. C. 』这两个！
19. 选择预设语系==>由于我不使用 X-Windows ，而终端机界面（纯文字界面的情况下）并没有办法提供中文的显示，选择中文作为预设的语系反而会在纯文字界面下出现乱码！！所以我都是选择『 English 』作为我的预设语系啰！
20. 选择时区==>在台湾，当然选择『 Asia/Taipei 』啰！
21. 设定密码==>这里要特别告诫大家，密码最好『多于八个字符』，并且含有『非英文或数字的特殊符号』为较佳的选择！当然啰，你也不能忘记呀！选择 Password 会有两次，提供你输入正确的密码！
22. 设定使用者 ID ==>我这里通常都先不设定的！所以就跳过去吧！先不设定啰！
23. 选择加密的条件==>也使用默认值就可以了！所以按下『 tab 』键移动至『 OK 』后，按下 Enter 吧！

24. 套件选择==>呵呵！终于来到重头戏了！底下提供三个选择，你可以参考看看！通常，如果你的硬盘很大的话，那么将光标移动到最底下，选择『 Every thing 』来个完全安装即可！但是为了安全性，不建议选择 Every Thing 啦！VBird 建议以『建议二』的方式来安装，不过，如果你的硬盘很小的话，那么就选择『建议一』来安装吧！无论如何，VBird 的系统中是以『 建议二 』安装的！

建议一	建议二	建议三
o Network Support o Dialup Support o Messaging and Web Tools o Router/Firewall o Network/Managed Workstation o Utilities	o Network Support o Dialup Support o Messaging and Web Tools o Router/Firewall o Network/Managed Workstation o Utilities o Software Development o Kernel Development	o 选择『 Every Thing 』
全部套件共占 366 MB	全部套件共占 657 MB	全部套件共占 2902 MB
适合只想要作为 NAT 之用的机器	可以后续再加入套件！	新手的安装啦！

25. 大概就这样吧！若想要跟 VBird 一样的系统，那就直接以『建议二』安装啰！
26. 开始正式安装啰==>这个时候系统会跟你说：『安装的过程中，会将信息都记录在 /var/install.tmp 档案中』不理他，直接给他『 OK 』下去！然后系统就会正式的作底下的工作啰：

    1. 先 Formating ，这个动作蛮快的，但是如果你前面选择了『 check bad blocks 』那么可能就会花很长一段时间！
    2. 再来是 Copying files ，然后是：
    3. Package Installation ！通常是蛮快的，由于我们安装的套件并不多！不过，如果你是选择 Every Thing 的话，那么可能要花比较长的时间喔！过程中需要抽换光盘片喔！

27. 制作开机片==>Boot Disk 最好作一下吧！有备而无患呀！
28. 完成安装==>屏幕显示出 Complete 的时候，哈哈！恭喜你啦！这样就 OK 啰！

注意事项：

- 你可能会觉得很奇怪，为什么你的安装过程会跟我的不一样？！呵呵！由于 VBird 的安装是比较简易的，包括我也没有安装 X-Windows 呀！所以当然也就没有 X-Windows 的测

试的画面啰！因此，如果你再安装的过程中选择了跟我不一样的套件，不用担心，安装过程会有些许的不相同的！

- 在安装完成之后，请千万记得『取出光盘片』，不然又会在进入一次安装画面喔！
- 同时建议，安装完成之后，请进入您的 BIOS 当中，将开机的顺序改回来『 C、A 』或『 C only 』反正就是让硬盘开机啦！

好了！这样应该就已经安装完毕了！请继续往下看看吧！而且，相当的建议您，在正式的进行架站之前，请依序看一下底下的网页，最好不要跳着看，不然的话，嘿嘿嘿嘿！出现什么问题可不要怪我！因为，照着顺序看会对你的 Linux 认识比较有帮助啦！

建立软盘开机片

建立软盘开机片一直是个好主意！他可以在你求助无门的时候给你莫大的帮助喔！所以，建立一个新的软盘开机片是一个好主意啦！如何建立呢？其实真的是很简单，不过，需要你的系统核心的版本就是了！依序进行底下的步骤（先将软盘片塞进软盘机中喔！）

```
[root @tsai / ]# uname -r
2.4.7-10                                  <==先取得核心的版本
[root @tsai / ]# mkbootdisk --device /dev/fd0 2.4.7-10
Insert a disk in /dev/fd0. Any information on the disk will be lost.
Press <Enter> to continue or ^C to abort:  按下 enter 吧！
```

注意一下上表的第三行，mkbootdisk 是制作开机软盘的指令，而 /dev/fd0 是软盘的代号，至于 2.4.7-10 则是我们系统的核心。要注意的是，如果你的核心曾经更新过，那么你的核心将不会是预设的 2.4.7-10 喔！需要跟着改变才行！这样就制作 OK 啰！然后将你的可开机软盘贴上卷标，给他保存起来吧！

本章习题练习(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- Linux 的目录配置以『树状目录』来配置，至于磁盘分割区 (partition) 则需要与树状目录相配合！请问，在预设的情况下，在安装的时候系统会要求你一定要分割出来的两个 Partition 为何？
  答：

  就是根目录『/』与虚拟内存『Swap』

- 什么是 IDE 界面，一般而言，普通 PC 允许几个 IDE 界面与装置？
  答：

  IDE 为用来传输硬盘数据的一个汇流界面；
  共有 IDE1, IDE2 ，分别有 master 与 slave 所以共四个 IDE 装置支持！

- IDE2 的 master 之第一个 logical 磁盘中，其装置代号（文件名称）为何？
  答：

  /dev/hdc5

- 在硬盘分割 (Partition)时，最多有几个 primary + extended ？
  答：

  Primary + Extended 共四个，其中 Extended 只有一个！ (更详细的硬盘与 MBR 可以参考 这里 这篇讨论)

- 若在分割的时候，在 IDE1 的 slave 硬盘中，分割『六个有用』的扇区（具有 filesystem 的），此外，有两个 primary 的扇区！请问六个扇区的代号？
  答：

  /dev/hdb1(primary)
  /dev/hdb2(primary)
  /dev/hdb3(extended)
  /dev/hda5(logical 底下皆为 logical)
  /dev/hda6
  /dev/hda7
  /dev/hda8
  请注意，5-8 这四个 logical 相加的总和为 3!

- 一般而言，在 RAM 为 64MB 或 128 MB 的系统中， swap 要开多大？
  答：

  Swap 可以简单的想成是虚拟内存，通常他的建议大小为 RAM 的两倍，但是实际上还是得视您的主机规格配备与用途而定。约两倍的 RAM ，亦即为 128 MB 或 256 MB ，可获得较佳效能!

- 什么是 GMT 时间？台北时间差几个钟头？
  答：

  GMT 时间指的是格林威治时间，为标准的时间，而台北时间较 GMT 快了 8 小时!

- Tap, SCSI 硬盘, RAID, printer 的装置代号？
  答：

  Tap : /dev/ht0 (IDE), /dev/st0 (SCSI);
  SCSI H.D.: /dev/sd[a-p],
  RAID : /dev/md[0-15];
  printer : /dev/lp[0-2]

- 如果我的磁盘分割时，设定了四个 Primary 扇区，但是磁盘还有空间，请问我还能不能使用这些空间？
  答：

  不行！因为最多只有四个 Primary 的磁盘分割槽，没有多的可以进行分割了！且由于没有 Extended ，所以自然不能再使用 Logical 分割说

- 我的 Mandrake 9.0 在安装的时候，进行 X-Window 的测试时都不会成功，要怎么办呢？
  答： X-Window System 的！万一还是没有办法登入 X-Window 的话，没有关系！不要害怕！等到后来『系统管理员篇』的时候，我们再来`入的谈一谈 X-Window 的设定吧！！ ^_^。而，如果万一不幸不小心按下了测试,要怎么办呢？屏幕已经一片漆黑了!@_@,没关系,此时可以按下 [Ctrl] + [Alt] + [F1] 就可以回到原先的画面啦！

- 通常在安装 Linux 的时候，最重要的就是磁盘分割了！请问：磁盘分割通常要分成几个步骤？
  答：

  1. 进行磁盘分割 partition ；
  2. 进行格式化 format ；

- 磁盘分割之后会有所谓的 Primary, Extended 与 Logical 的磁盘分割槽，请问何者为可使用的 Partition ？
  答：

  只有 Primary 与 Logical 为可用， Extended 为不可直接使用的 Partition ，还需要再次的分割成为 Logical 之后，才可以继续使用！而最大可分割出来的 Partition 应该有 64 个才对！

安装 FC4 与多重开机小技巧

最近更新日期：2005/11/08

等了好久！终于要开始来安装我们的 Linux 练习机了！注意喔！ 既然这里特别强调的是『练习机』，所以，里面的种种建议都是『练习用』的喔！ 而且，这部练习机在您顺利的了解 Linux 的主机操作之前，『最好不要连上 Internet 』呢！很容易被入侵啊！ 我们这里使用目前最新的 Fedora Core Release IV 来进行安装喔！

Linux 安装的第一步『规划』

由 上一章的 Linux 主机规划 我们大致上可以了解：『Linux 主机的配备要求与 Linux 主机所提供的服务有关』，所以啰，要安装 Linux 的第一步就是先进行 Linux 主机的未来蓝图规划啦！那么该如何规划？由前一章的内容，我们不难了解，要规划好一个 Linux 主机的话，需要：

1. 决定 Linux 主机的角色定位;
2. 根据步骤一，选择适当的硬件配备;
3. 根据步骤一，决定实体硬盘的分割状态;
4. 根据主机上面的操作系统，选择适当的开机加载程序 (boot loader) ， 以便安装在主要开机扇区中 (Master Boot Recorder, MBR);
5. 根据步骤一，选择所需要的 Linux 程序套件;

好了，既然已经知道大概需要规划的原理之后，底下我们就实地的来操练一遍吧！好让您不会很快的忘掉去！

一个练习机的规划

假设：这部主机主要是为了我自己练习 Linux 之用， 不过，未来可能会接触到网络的实际练习。而为了方便数据查询，因此，我需要 X Window 系统的支持， 而且由于不熟悉 Linux ，因此也需要预留硬盘空间作为日后练习之用。如果是这样的状态， 您应该要如何规划你的主机呢？

1. Linux 主机角色的定位

由上面的假设状态，您可以知道这部主机主要是作为练习之用，需要 X Window ， 而且由于操作者是新手，因此，建议将所有的套件都安装上去呢！

2. 硬件的配备选择：

因为我使用 Linux 来作为练习，而且含有 X Window 在里面，预计是将所有的套件都装上去的，因此，硬盘不能少，CPU/显示卡等等的等级也必须要适中，不可用等级太低的。而且， 内存也应该大于 256MB 以上比较妥当。我使用的练习机设备为：

- 主机板与 CPU： 我的练习机 CPU 是 AMD 的 AthlonXP 1800+，搭配 Asus 的 A7V8X 主机板。
- 内存：我安插了 256MB 的内存。
- 硬盘：使用一颗淘汰下来的 20GB 硬盘。因为我只有一部主机，为了要同时使用 Windows 与 Linux 操作系统，因此，我选择了抽取式硬盘盒来帮忙抽换。 这样一来，比较大的优点是可以让我完全掌控一部主机，而不需要担心将 Windows 的系统删除。
- 网络卡：用最阳春的螃蟹卡(Realtek 8139 芯片)。
- 显示卡：这个重要，因为我要使用 X Window ，因此， 我用的是 32MB 的 ATI 的 9200 芯片组(现在这个等级已经算很差的配备了)。
- 其它的输出/输入装置：一部 DVD 光驱、pc104 键盘、光学鼠标、300W 电源供应器及 19 吋的传统映像管屏幕。

一般来说，这样的配备算是前一阵子的入门级机种了！目前的入门级机种都比这个配备好太多！您也可以利用您的 CPU 等级为 P-III 的主机来进行练习啊！ 但不建议使用 P-II 以下等级的机种。除非您不要使用 X Window ！请参考前一章的介绍。

3. 关于硬盘分割的分配：

因为是练习机，建议您主要分为 / 与 Swap 两个即可。不过，还是需要留下一些硬盘空间来作为练习之用。 我的 8GB 硬盘主要分为 6GB 为 / ， 512MB 为 swap ， 1GB 为 /home ，另外的磁盘空间则不规划！。 至于磁盘分割与各目录的相关意义，我们会在 磁盘档案系统 当中来进行说明，切莫着急啊！ ^_^。至于如果您的硬盘大于 60GB 以上时，可能会有开机扇区的问题，那就必须要独立出 /boot 这个磁盘槽了，请参考 本章最后一节的说明。

4. 选择 MBR 当中的开机管理程序：

在 Linux 里面预设使用两种开机管理程序，分别是 LILO 与 GRUB 这两个好东西，其中，LILO 算是比较早期的开机管理程序，不过， 鸟哥个人倒是还蛮喜欢 LILO 的， 毕竟他虽然比较死板，但是还蛮好用的，磁盘代号设定上面与 Linux 上的磁盘代号相同， 所以没有什么太大的困扰。而较新的 GRUB 其实是很棒的一套开机管理程序，我个人认为， 他最大的功能也最具魅力的地方是具有『动态搜寻核心档案』的功能， 他可以让您在开机的时候，可以自行编辑您的开机设定系统档案，呵呵！所以即使您不小心设定错了 grub ，没关系！开机的时候自行编辑一下就好啦！这方面的技巧， 我们会在 开机流程与 Loader 的时候再来详细的介绍， 还是慢慢的从头学习起来呦！

5. 选择所需的套件：

   虽然将光盘上面的全部套件都安装，是有点浪费硬盘空间，不过我们是练习机嘛！ 所以还是完整的都给他安装下去的好。(如果您想要选择套件的话， 特别建议您，务必将『系统开发工具』，也就是 gcc, kernel-headers, kernel-source 等等安装上去喔！)

到了这一步之后，嘿嘿！规划就已经差不多了，所以，这个时候，基本上已经可以开始来安装 Linux 啦！但是，还是有个困扰耶，那就是，在第三步骤的时候，我要怎么在安装的时候分割我的硬盘呀！？ 上一章里面有提过硬盘的排线与硬盘在 Linux 里面的磁盘代号有关，那么该如何分割？ 另外，有什么自订的方式可以来帮我分割硬盘吗？呵呵！底下我们就来提一提如何进行这些动作啰！

硬盘的连接与代号

硬盘分割与配置的好坏，会影响到未来您的主机的使用情况，此外，好一点的分割方式， 会让您的数据保有一定的安全性！怎么说呢？这么想好了，如果你的 Windows 硬盘里面，仅有 C 槽的话，那么当 Windows 需要重新安装的时候，你又想要重新格式化 (format) 时，而 C 槽里面很不巧的，已经放了很多重要的档案数据，这个时候怎么办？ 光是搬这些重要数据到其它空间就受不了！ 所以，比较聪明的玩家，都喜欢分割成两槽以上，将系统档案与数据文件分开， 可以达到比较好的管理效果！除此之外，磁盘分割的好坏，还可以影响到系统存取数据的效能呢！ 这个部分我们在后面几章再来谈一谈！

所以啰，正常使用情况下的 Linux 主机，通常会依照目录与主机的特性，来分割硬盘， 以达到比较好的管理成效。不过，由于 Linux 的硬盘分割比较具有弹性，同时， Linux 硬盘分割程序 fdisk 功能很强悍，此外，要分割的好，必须要了解一下基础的硬盘架构， 所以，底下我们先来介绍一下硬盘的基本架构，然后再来介绍如何分割吧！

现在的主流硬盘应该是 SATA 小排线的那一种接口硬盘。他与旧 IDE 硬盘的分别， 我们已经在前一章谈过了，这里就略过不提。但不论是 SATA 或者是 IDE 接口的硬盘， 他在 Linux 当中的硬盘代号都是一样的！所以，这里我们依旧以 IDE 硬盘来稍作说明啰！

通常在 586 之后生产的主机板上面都有两条接排线的界面 (排线就是用来连接硬盘与主机板的那一个东西啦！)，而我们称这种界面为 IDE 界面，这也是之前的主流硬盘界面(目前已被 SATA 取代) ，为了区隔硬盘读取的先后顺序，所以主机板上面的这两个界面就分别被称为 Primary (主要的) 与 Secondary (次要的) IDE 接口啰， 或者被称为 IDE1 (Primary) 与 IDE2 (Secondary)。

而如果你有仔细观察的话，那么每一条排线上面还有两个插孔，也就是说一条排线可以接两个 IDE 界面的装置 (硬盘或光驱)，而你有两条排线，因此 一个主机板在预设的情况中，应该都可以接四个 IDE 界面的装置。好了，那么每条排线上面该如何判别哪一个是主硬盘 (Master)，哪一个是副硬盘 (Slave) 呢？这个时候就需要调整硬盘上面的跳针 (jump) 才可以知道！请察看一下您的硬盘机吧！上面应该都会有图示说明才对。 (注：硬盘的 master/slave 判断方法中，除了利用 jump 主动调整之外， 还可以透过 cable 自动选择。)

好了，所以如果我有一个光驱了，那么我最多就只能再安装三部 IDE 接口的硬盘在我的主机上面。OK！那么由于我的硬盘与 Linux 的磁盘代号有关，那么我怎么知道这个硬盘的代号呢？没问题啦，由 IDE 1

(Primary IDE) 的 Master 硬盘先计算，最后是 IDE 2 的 slave 硬盘，所以各个磁盘的代号是：

IDE\Jumper	Master	Slave
IDE1(Primary)	/dev/hda	/dev/hdb
IDE2(Secondary)	/dev/hdc	/dev/hdd

假如我只有一颗硬盘，而且这一颗硬盘接在 IDE 2 的 Master 上面，那么他在 Linux 里面的代号就是 /dev/hdc 啰！OK！好像没问题了呦！呵呵！才不是呢，问题很大呦！ 因为，如果我这个磁盘被分割成两个磁盘分割槽(Partition)，那么每一槽在 Linux 里面的代号又是如何？如何知道每个 partition 的代号呢？

认识硬盘的 partition

基本上，硬盘是由最小的物理组成单位 扇区 (sector) 所组成的，而数个扇区组成一个同心圆时，那就称为 磁柱 (cylinder) ，最后构成整个硬盘的容量大小。关于硬盘的管理我们在后续章节再来介绍， 这里我们比较想要知道的是，如何分割硬盘，所以先简单的将硬盘变成如下的图标：



图 1、硬盘数据示意图

在上面的图示中，我们可以很清楚的知道，在硬盘里面有分为两个区域，一个是放置这个硬盘的信息区，我们称为 Master Boot Recorder, MBR (主要开机扇区)， 一个则是实际档案数据放置的地方。MBR 可以说是整个硬盘最重要的地方了，因为在 MBR 里面记录了两个重要的东西，分别是：开机管理程序，与磁盘分割表 (partition table)。因此，只要 MBR 物理实体坏掉了，那么这颗硬盘就差不多要报废了！ 因为，如果系统找不到 partition table ，就无法使用这块硬盘，所以数据即使没有丢掉，但是没有 MBR ，呵呵，还是不能使用的啦！

首先来看一看什么是 partition table 呢？简单的说，我们说的『 硬盘分割』就是在修改这个 partition table 而已！他基本上定义了『第 n 个磁盘区块是由第 x 磁柱到第 y 个磁柱』，所以，每次当系统要去读取 n 磁盘区块时，就只会去读取第 x 到 y 个扇区之间的数据！呵呵！这样知道了吗？很简单吧！下次记得人家在谈磁盘分割的时候， 不要以为系统真的会在硬盘上面用力、努力的划标签！实际上，他最大的功能就是修改 MBR 里面的 partition table 啦！

不过，由于这个 MBR 区块的容量有限，所以，当初设计的时候，就 只有设计成 4 个分割纪录，这些分割记录就被称为 Primary (主分割) 及 Extended (延伸分割) ，也就是说，一颗硬盘最多可以有 4 个 (Primary + Extended) 的扇区，其中， Extended 只能有一个， 因此，你如果要分割成四块磁盘分割的话，那么最多就是可以：

P ＋ P ＋ P ＋ P
P ＋ P ＋ P ＋ E

的情况来分割了。其中需要特别留意的是，如果上面的情况中， 3P + E 只有三个『可用』的磁盘，如果要四个都『可用』，就得分割成 4P 了！(因为 Extended 不能直接被使用，还需要分割成 Logical 才行，底下我们会继续说明的！)。那么为什么要有 Extended 呢？这是因为如果我们要将硬盘分割成 5 个区块的话，那么怎么办？这个时候就需要 Extended 的帮忙了。

由于 MBR 仅能保有四个 partition 的数据记录，那如果超过 4 个以上时， 系统允许在额外的硬盘空间放置另一份磁盘分割信息，那就是 Extended 了！ 假设您将您的硬盘分割成为 3P + E ，那么那个 E 其实是告诉系统，磁盘分割表在另外的那份 partition table， 也就是说，那个 Extended 其实就是具有『指向 (point)』正确的那个额外的 partition table 啦！ 本身 Extended 是不能在任何系统上面被使用的，还需要再额外的将 Extended 分割成 Logical (逻辑) 分割才能被使用，所以啰，藉由这个 Extended 的帮忙，我们就可以分割超过 5 个可以利用的 partition 啰！ 不过，在实际的分割时，还是容易出现问题的，底下我们来思考看看：

- 思考一：如果我要将我的大硬盘『暂时』分割成四个 partition ，同时，还有其它的空间可以让我在未来的时候进行规划，那么该如何分割？

  说明：

  由刚刚的说明，我们可以知道， Primary + Extended 最多只能有四个 partition，而如果要超过 5 个 partition 的话，那么就需要 Extended 的帮忙。因此，在这个例子中，我们『千万不能分割成四个 Primary 』为什么呢？假如您是一个 20 GB 的硬盘，而 4 个 primary 共用去了 15 GB ，您心想还有 5 GB 可以利用对吧？错！剩下的 5 GB 『完全不能使用』，这是因为已经没有多余的 partition table 纪录区可以记录了，因此也就无法进行额外的分割，当然啰， 空间也就被浪费掉了!因此,请千万注意,如果您要分割超过 4 槽以上时,请记得一定要有 Extended 分割区，而且必须将所有剩下的空间都分配给 Extended ，然后再以 logical 的分割区来规划 Extended 的空间。 另外，考虑到磁盘的连续性，一般建议将 Extended 的扇区分割在最后面的磁柱内。

- 思考二：我可不可以仅分割 1 个 Primary 与 1 个 Extended 呢？

  说明：

  当然可以！基本上， Logical 的号码可达 63 号，因此， 你可以仅分割一个主分割，并且将所有其它的分割都给 Extended ，利用 Logical 分割来进行其它的 partition 规划即可！

- 思考三：假如我的硬盘安装在 IDE 1 的 Master ，并且我想要分割成 6 个可以使用的硬盘扇区，那么每个磁盘在 Linux 底下的代号为何？

  说明：

  由于硬盘在 Primary ＋ Extended 最多可以有四个，因此，在 Linux 底下，已经将 partition table 1 ~ 4 先留下来了，如果只用了 2 个 P + E 的话，那么将会空出两个 partition number 呦！再详细的说明一下，假设我将四个 P + E 都用完了， 那么硬盘的实际分割会如同下图所示：

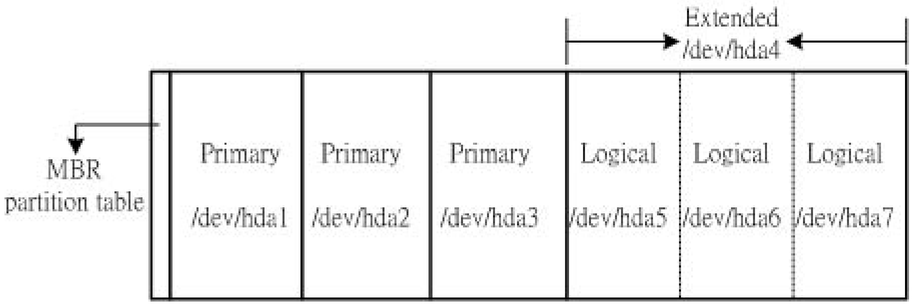


图 2、六个硬盘的分割示意图

实际可以使用的是 /dev/hda1, /dev/hda2, /dev/hda3, /dev/hda5, /dev/hda6, /dev/hda7 这六个 partition！至于 /dev/hda4 这个 Extended 扇区本身仅是用来规划出让 Logical 可以利用的磁盘空间而已！(其实在每个 partition 的最前面扇区，会有一个特殊的区块， 称为 super block，我们的 Extended 指向的，就是 /dev/hda4 的 super block 处， 该处就是额外记录的那个 partition table 啦！)

那么万一我只想要分割 1 个 Primary 与 1 个 Extended 呢？这个时候你的磁盘分割会变成如下所示：

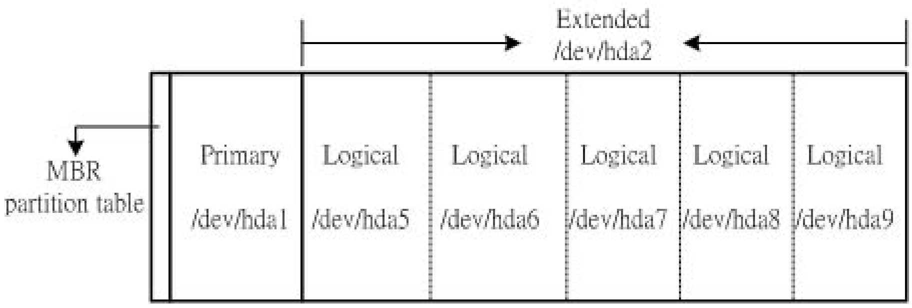


图 3、六个硬盘的分割示意图

注意到了吗？因为 1~4 号已经被预留下来了，所以第一个 Logical 的代号由 5 号开始计算起来，而后面在被规划的， 就以累加的方式增加磁盘代号啰！而其中 /dev/hda3, /dev/hda4 这两个代号则是空的， 被保留下来的代号。

Linux 安装模式下，硬盘分割的选择(极重要)

实际上，在 Linux 安装的时候，已经提供了相当多的预设模式让您选择分割的方式了， 不过，无论如何，分割的行为都不是很能符合自己主机的样子！因为毕竟每个人的『想法』都不太一样！ 因此，强烈建议使用『自订安装, Custom 』这个安装模式！ 在某些 Linux distribution 中，会将这个模式写的很厉害，叫做是『 Expert, 专家模式 』，这个就厉害了，请相信您自己，了解上面就自称为 专家 了吧！没有问题！

- 自订安装『Custom』：

- A：初次接触 Linux ：只要切割『 / 』及『 Swap 』即可！

  好了，通常初次安装 Linux 系统的朋友们，我们都会建议他直接以一个最大的扇区『 / 』来安装，这样有个好处，就是不怕分割错误造成无法安装的困境！例如 /usr/ 是 Linux 的可执行程序及相关的文件摆放的目录，所以他的容量需求蛮大的， 万一你分割了一块扇区给 /usr ，但是却给的不够大，那么就伤脑筋了！ 因为会造成无法将数据完全写入的问题，就有可能会无法安装啦！因此上， 如果你是初次安装的话，那么可以仅分割成两个扇区『 / 与 Swap 』即可！

- B：建议分割的方法：预留一个备份的扇区！

  就如同前面几个心得分享文章中提到的，由于 Linux 预设的目录是固定的，所以，通常我们会将 /var 及 /home 这两个目录稍微加大一些，如果硬盘够大的话，加个几 GB 也不为过！另外， /usr 至少给他 3~5 GB 吧，如果硬盘真的大的话！而 / 也可以给个几 GB 的空间。最后，由于我们的 Linux 可能是在『试用』阶段，所以很有可能会重复的一再安装，因此上，鸟哥 都会预留一个扇区来备份我的核心啦与实作过程中觉得不错的 scripts（ 就有点像 DOS 的批次档 ），当然，我的 /home 底下的咚咚也可以有备份的地方， 而安装套件的源文件也可以摆在这里！有个最大的好处是，当我的 Linux 重新安装的时候，我的一些套件马上就可以直接在硬盘当中找到！ 呵呵！重新安装比较便利啦！

- 选择 Linux 安装程序提供的预设硬盘分割方式：

对于首次接触 Linux 的朋友们，通常不建议使用各个 distribution 所提供预设的 Server 安装方式，因为会让你无法得知 Linux 在搞什么鬼，而且也不见得可以符合你的需求！注意：选择 Server 的时候，请『确定』您的硬盘数据是不要的！因为 Linux 会自动的把你的硬盘里面旧有的数据全部杀掉！此外，硬盘至少需要 2 GB 以上才可以选择这一个模式！
硬盘方面的规划大致上就是如此啦！要规划硬盘的时候，请特别的小心呦！

开始安装 Fedora Core Release IV

Linux 安装之前要准备什么呢？就是刚刚前面已经讲过的几个咚咚啦！归纳一下：

1. Linux 主机规划单：就是刚刚我们规划好的那个数据啰！
2. Linux distribution ：利用一些映象站台下载各版本的 Linux ，或者直接以本书提供的四块 CD 装的 FC4 进行安装啰！
3. 主机硬件信息收集：根据主机规划单的内容， 去收集一下你的硬件信息吧！其中特别重要的是，先检查一下是否可以使用光盘开机呢？如果 BIOS 不能支持光盘开机的话，那么就需要先行安装可开机软盘。
4. 网络硬件联机：这部份本书先不谈，否则内容就太多了， 阿！再写下去鸟哥会疯掉……所以请大家先上网查阅一下网络的硬件联机吧！
5. 网络信息：包括你的 IP, netmask, gateway, dns IP 、是否为拨接等等，都需要先知道呦！

然后，其实各个套件的安装步骤都差不多，大概都是：

- A. 选择安装模式：主要分为图形接口安装与文字接口安装； 如果是图形接口安装的话，还可以选择语系，这个时候我们就有中文可以使用啦！

- B. 搜寻硬件信息：然后安装程序会去搜寻一下系统的硬设备， 以利后续的处理，有的安装程序会在这个地方让您加入一些参数，以驱动不明的装置设备；
- C. 设定键盘、鼠标模式：这个可是很重要的项目呀！
- D. 硬盘分割设定：就是刚刚提到的几个注意事项；
- E. 套件选择：这是很重要的部分呢！请特别留意！
- F. 网络与安全性设定：连上 Internet 的模式与驱动网络卡的方式等设定；
- G. 超级管理员与一般身份使用者账号设定：最重要的是设定 root （ 超级管理员 ） 的密码啰！
- H. 设定 X-Window 与开机片：如果有安装 X-Window 相关的软件，那么 X-Window 就需要设定并且测试一下！

大概就是这样子吧！好了，底下我们就真的要来安装啰！

---

1. 选择开机次序并开机：

因为目前几乎所有的 Linux Distributions 都是支持光盘开机的， 而我们的主机板也几乎都是支持光盘开机。只是，您必须要确定系统的第一个开机搜寻装置为光驱就是了。 我们可以在 BIOS 里面设定开机的次序，看看能不能以光驱开机！ 设定方式为：

1. 按电源键开机；
2. 在进入系统之前会出现 Del 字样（每个厂牌不太相同），此时按下键盘上的 Delete 键；
3. 进入 BIOS 之后以方向键选择 『BIOS Features Setup』这一项，或者是 『Advanced BIOS Features』，不管如何，反正只要看到 『BIOS Features』字样的那一项就对了！；
4. 将方向键移动至 『Boot Sequence』 或者是 『First Boot Device』； 这一项，按键盘上的『Page Up』 或 『Page Down』 按键，选择『CD-ROM』 为第一开机顺位即可。这里注意一下，如果你的机器并不支持 CD-ROM 开机的话，你一定找不到 CD-ROM 这一项。这就比较麻烦，因为目前有些 Linux distributions 仅支持光盘开机的说～所以，您就得要找比较早期的版本或者其它方式来安装了。
5. 按键盘上 『ESC』 键退出；
6. 将方向键移动至 『Save and Exit』 这一项按 『Enter』 及 『Y』 确认后重新开机即可！

在进行完上面的步骤之后，请将第一片 Fedora Core IV 可开机光盘放入光驱中，按下电源，给他开机去！

Tips:
其实，目前除了一般的光盘片之外，很多版本的 Linux 也提供可开机 DVD 片了！ 所以，如果您嫌 4 片装的 FC4 太麻烦，那么可以到义守大学下载 DVD 版本的 FC4 喔！
http://ftp.isu.edu.tw/pub/Linux/Fedora/linux/core/4/i386/iso/
另外，那个 FC4-i386-rescuecd.iso 档案也可以下载！那个是救援光盘～ 如果发生不可预期的错误时，利用这个光盘可以救回来您的 FC4 喔！



---

2. 选择安装模式：

在进行完上面的动作之后，理论上，您的主机已经以 FC4 可开机光盘开机成功了！ 如果发生一些错误讯息时，很可能是由于： 1)硬件不支持； 2)光驱会挑片； 3)光盘片有问题； 如果是这样，那么建议您，

再仔细的确认一下您的硬件是否有超频？或者其它不正常的现象。 而，您的光盘来源也需要再次的确认！(如果是书上附赠的光盘，却发现无法开机成功,先确定一下您的光驱是否会挑片？换一台光驱试看看。如果还是无法开机，可以寄回您的书商，请他们帮忙您换一组光盘呢！这是您的权利喔！)

一般 Linux 都会支持至少两种安装以上的安装模式,分别是文字(text)与图形(graphic)接口。正常的话，以光盘开机后，会出现如下图示。



图 4 、选择安装模式

如果想要以图形接口来安装，可以直接按下 <enter> 按键，如果想要以文字接口来安装， 可以在箭头指的地方输入『 linux text 』来让安装程序以文字接口安装。 不过，要注意的是，如果在 10 秒左右您没有在 boot: 后(箭头处)输入任何按键的话， 那么安装程序就会以预设的模式来安装，预设是以图形接口来安装的喔！我们这里也使用图形接口来介绍。 其实，两个接口都很像啦！只是图形接口还可以使用中文就是了！ ^_^

在按下 Enter 之后，安装程序就会开始去侦测硬件，并且将信息列在屏幕上给您参考， 如下图所示。由于鸟哥为了捉图的需要，所以使用 VMWare 之类的软件来仿真安装。 不过，就如同前面介绍的，不建议您用这样的软件来安装喔！这里只是作个介绍而已。 图 5 的箭头处就指出一些 IRQ 的利用，以及安装程序侦测到的硬件相关信息(例如 VMWare 仿真的硬盘！)

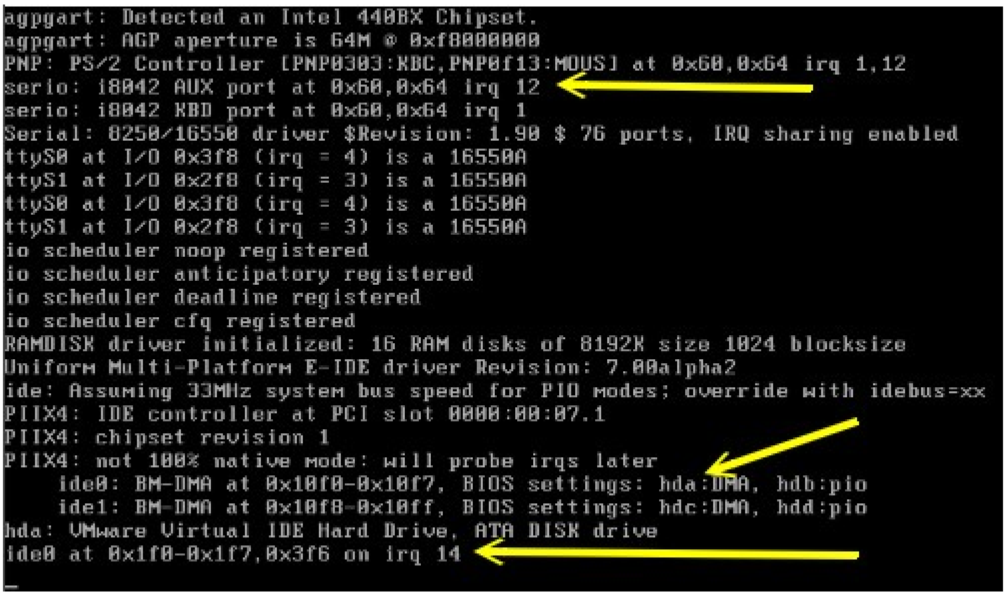


图 5 、进行硬件侦测过程

硬件侦测完之后，会出现一个是否检查光盘的画面，如下图所示。 注意，如果要检查光盘的话，会花去很多时间的！所以，如果确定光盘来源没有问题， 请选择『 Skip 』选项即可！



图 6 、是否检查光盘？请选择 skip 喔！

略过光盘检验工作后，因为我们使用的是图形接口的安装模式，所以安装程序就会去侦测： 屏幕、键盘、鼠标等等相关的硬件啰！如下图所示啊！

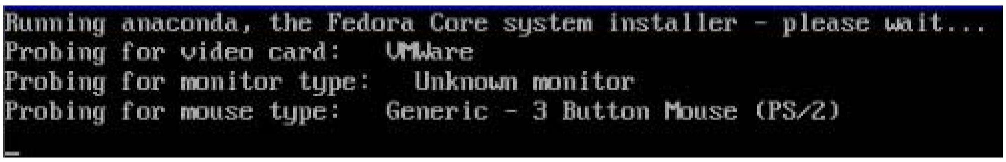


图 7 、安装程序侦测到的屏幕、显示卡与鼠标等信息

3. 选择安装程序的语系与键盘配置：

在完成了一些硬件方面的侦测之后，顺利的话，就可以进入图形接口的安装了！ 安装的画面如下图所示。基本上，分为左右两个区块，左边主要是作为『说明』之用， 右边才是真正的操作区块！如果您搞不懂这个安装画面是干嘛用的，可以参考左边区块的说明。 至于右下角则是下一步或者回到上一步的按钮喔！给他按下一步吧

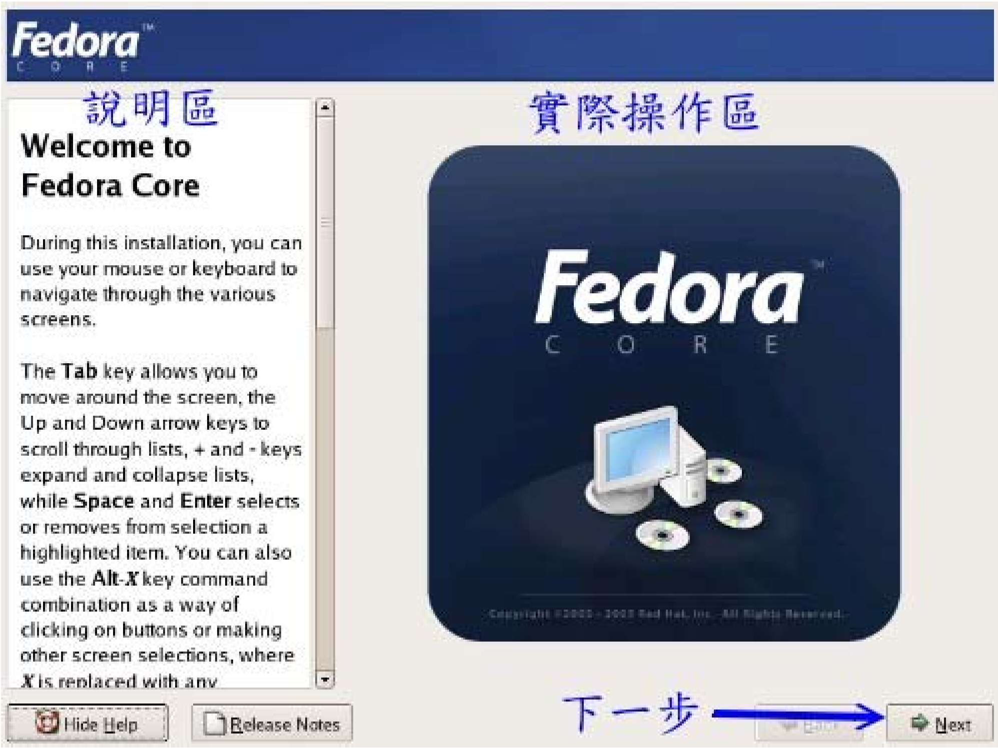


图 8 、FC4 的欢迎画面

之后就是选择语系啦！因为安装程序可以使用很多不同的语言来呈现。 我们可以选择中文来进行安装喔！如下图所示，先选择『繁体中文』按『下一步』即可！

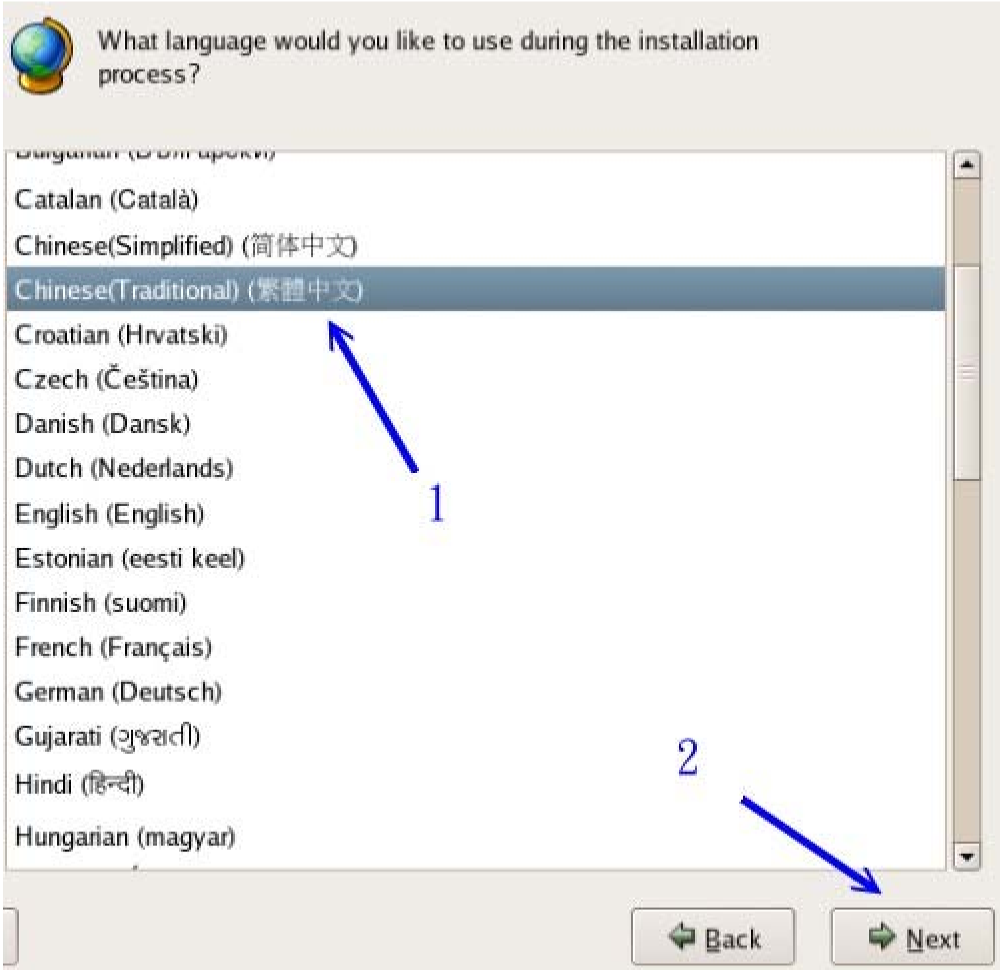


图 9 、语系的选择

嘿嘿！此时竟然是以中文来显示我们所需要的画面了！啊！真是太高兴了！ ^_^ 接下来，则要选择『键盘的配置』。因为每个地区的键盘上面的字母配置都不一样， 我们使用的是英文的键盘配置，所以，选择『美式英文』就可以了！如下图所示。

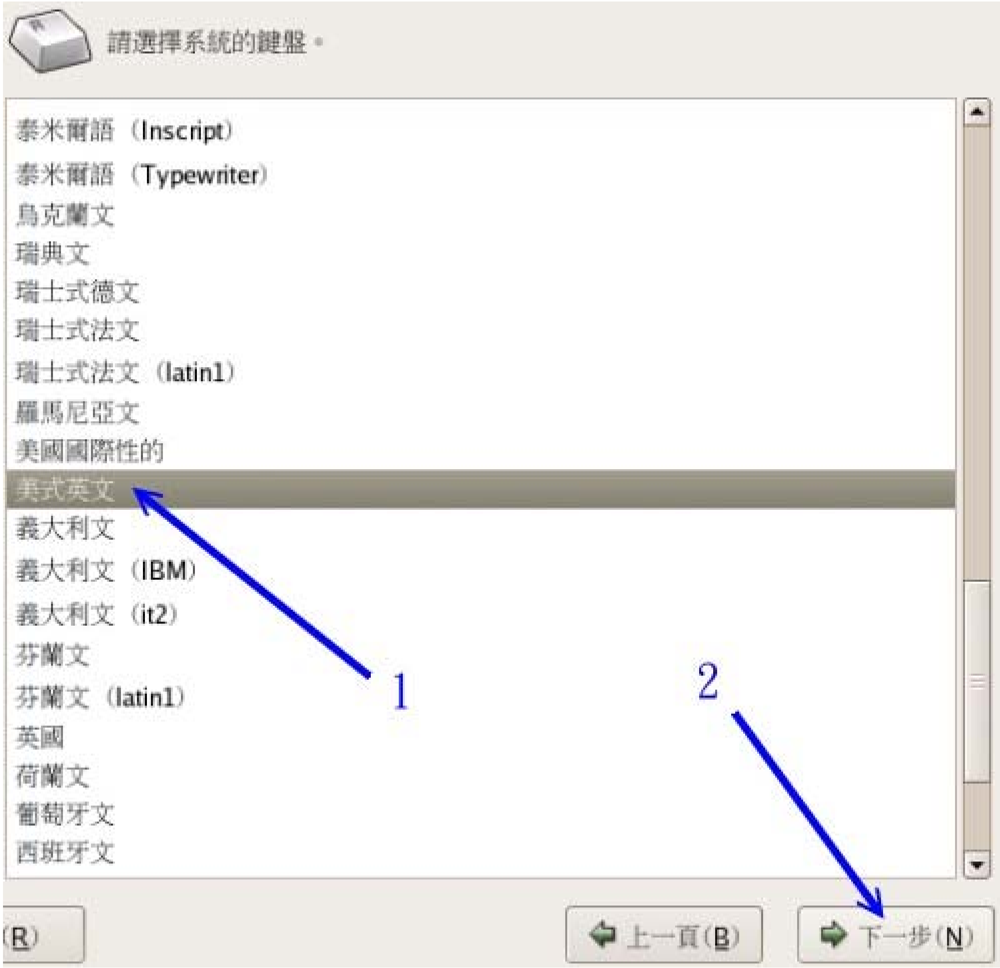


图 10 、键盘配置的选择

---

4. 安装的系统类型与磁盘分割：

什么是『安装的系统类型』呢？基本上， FC 4 已经帮您规划好一些主机利用的方式了。 举例来说，如果您想要使用桌上型计算机的功能，那么可以选择下图的『个人计算机』项目， 他会主动的帮您进行好磁盘分割以及相关的套件选择啊！不过，缺点是， 可能您的硬盘 partition 就交给系统主动去判断处理，在学习上，会比较不好， 而且，系统的预设分割与套件的选择，也不见得就会跟您想象的一样！ 因此，这里强烈的建议您，务必选择『自订安装』喔！

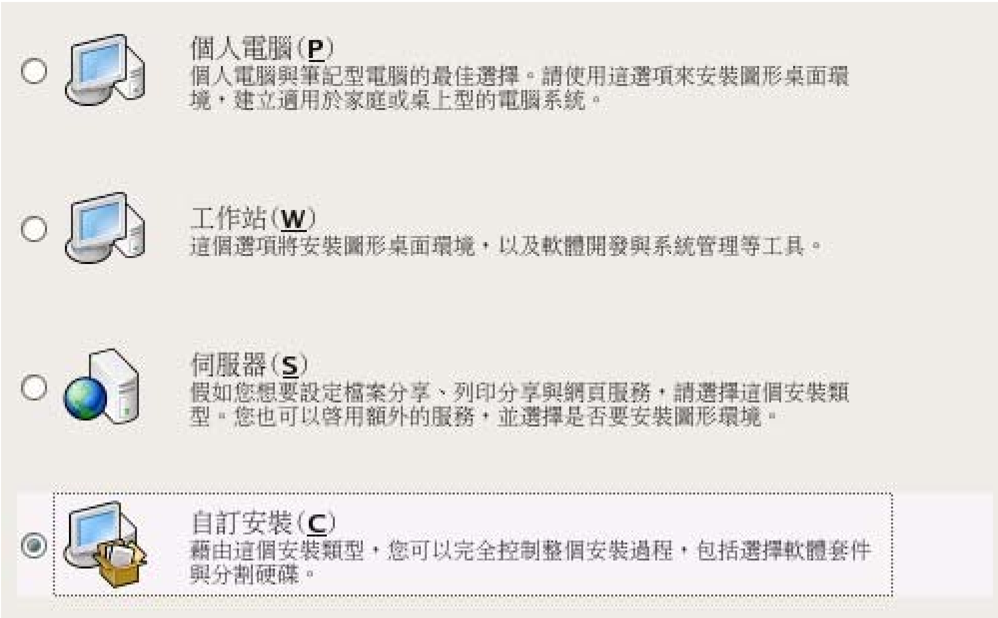


图 11 、预设的安装系统类型

好了，接下来自然就是要进行磁盘的分割动作了!这是我们在安装与规划的部分一直强调的地方呢! ^_^ 此时，请选择『使用 Disk Druid 』工具来自行进行磁盘分割呢! 这可是很重要的喔!



图 12 、磁盘分割的模式(自动/手动)

如果是一颗全新的硬盘，可能会发生如下的错误讯息，这个讯息仅是告知您， 安装程序找不到 partition table 而已，还不需要太担心啦! 此时， 直接按下『是』就可以了!

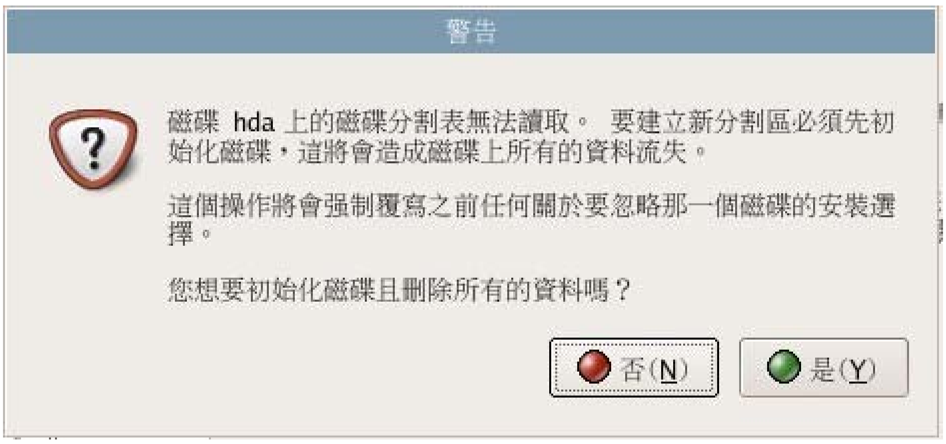


图 13 、确定是否删除 partition table

接下来的画面则是在操作磁盘分割的主要画面了！这个画面主要分为三大区块， 最上方为硬盘的分割示意图，目前因为我的硬盘并未分割，所以呈现的就是一整块而且为 Free 的字样。 中间则是指令区，下方则是每个分割槽(partitions)的启始磁柱、结束磁柱、所占容量大小， 以及相关的档案系统与挂载点了。关于挂载点我们会在后面几个章节加以介绍， 这里您只要知道我们要将磁盘分割槽挂载到 /home 以及 / 还有 swap 即可。

至于指令区，总共有六大区块，其中 RAID 与 LVM 是硬盘特殊的应用， 我们先略过不谈(在基础篇的最后面硬件维护的章节，我们会来谈一谈 LVM)。 指令的作用如下：

- 『新增』是增加新分割，亦即进行分割动作，以建立新的磁盘分割槽；
- 『编辑』则是编辑已经存在的磁盘分割槽，您可以在实际状态显示区点选想要修改的分割槽， 然后再点选『编辑』即可进行该分割槽的编辑动作。
- 『删除』则是删除一个磁盘分割槽，同样的，您得要再实际状态显示区点选想要删除的分割槽喔！
- 『重设』则是恢复最原始的磁盘分割状态！

需要注意的是，您的系统与鸟哥的系统当然不可能完全一样，所以，您的屏幕上的硬盘信息， 应该不会与鸟哥的相同的喔！所以看到不同，不要太紧张啊，那是正常的！



图 14 、磁盘分割的主画面

好了！准备来进行分割的动作吧！按下『新增』之后，会出现如下图的窗口～ 我们得选择挂载点，我首先规划出来的是根目录 / 的分割槽，所以先点选如下图最上方箭头指的地方， 然后选择 / 即可；

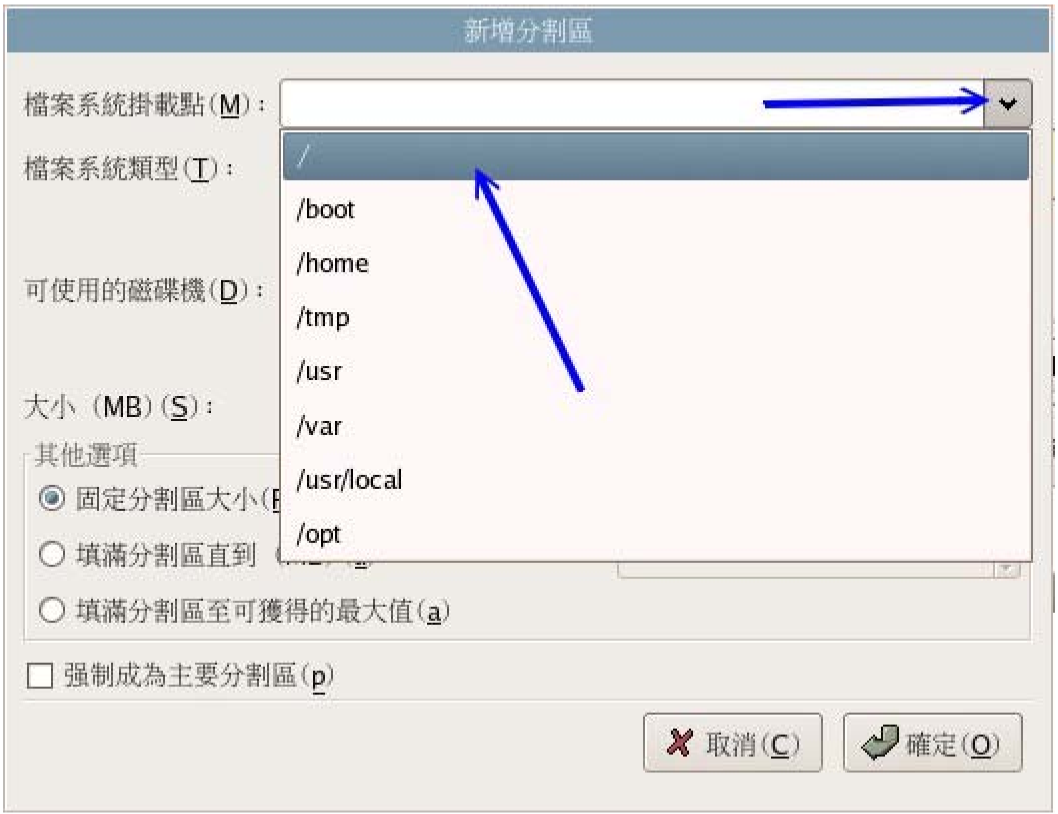


图 15 、选择根目录的磁盘分割

接下来，因为我仅有一颗硬盘，所以在『可使用磁盘驱动器』就没有办法选择，一定是固定的那一颗啦！ 而我想要的磁盘档案系统类型就选择标准的 ext3 即可!再来则是选择分割的状态了。首先,我要规划出 6GB 左右，因此，在『大小』那个地方填入 6000(大约是 6000M＝6G)。 然后我想要固定大小，因此选择『固定分割区大小』且选择『强制成为主要分割区』，这个就是 primary 的 partition 啦!最后按下确定。(注意，如果您想要将系统的所有套件安装 这里至少需要填入 10000 ，亦即 10GB 的空间才足够喔！)

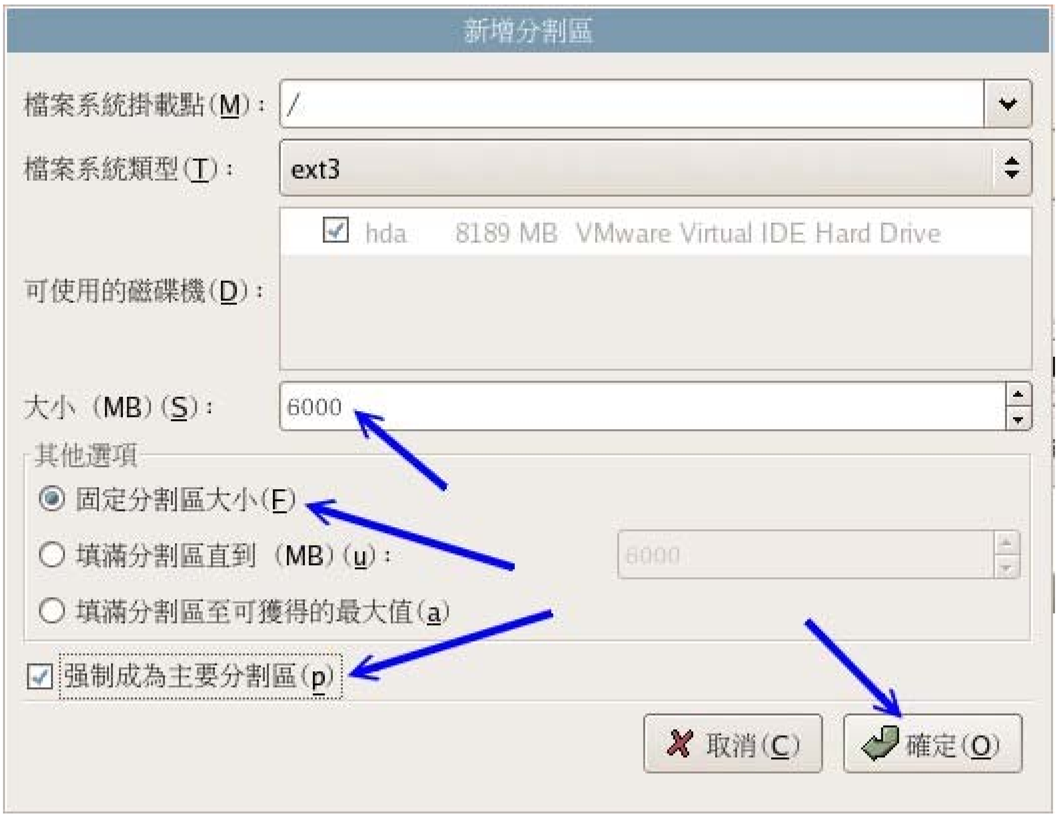


图 16 、选择根目录的磁盘分割

之后会回到主画面，就如下图所示，在『示意图』方面就已经有一个大区块出现了！ 在实际状态区，则有 /dev/hda1 出现了！很简单吧！好了，继续来进行虚拟内存 swap 的分割吧！



图 17 、已经分割出 / 的画面

同样的按下『新增』然后就会出现同样的画面，此时，我们直接在『档案系统类型』的地方， 选择『swap』这个类型即可！这个 Swap 有什么功能呢？简单的说，他可以被看做为 『虚拟内存』啰，那么虚拟内存是什么？ 您可以这样想象，当你的物理内存只有 64 MB 的时候，但是你的系统负荷突然之间太大了， 例如突然之间有数十个人连上你的 Web 服务器时，那么你的物理内存将不足以负荷这些计算的数据！ 怎么办？这个时候我们可以使用硬盘来仿真内存的数据存取，这个就是所谓的『虚拟内存』啰！ 不过，虚拟内存的速度会比较慢呦！

当有数据被存放在物理内存里面，但是这些数据又不是常被 CPU 所取用时， 那么这些不常被使用的程序将会被丢到虚拟内存当中， 而将速度较快的物理内存空间释放出来给真正需要的程序使用！ 这就是虚拟内存 的功效之一啦！通常 Swap 建议的值大约是『RAM 的两倍大』，但是这个因地制宜啦！像鸟哥的 Proxy 主机本身的内存就达到 1GB 了， 那么是否还需要虚拟内存呢？见仁见智啰！

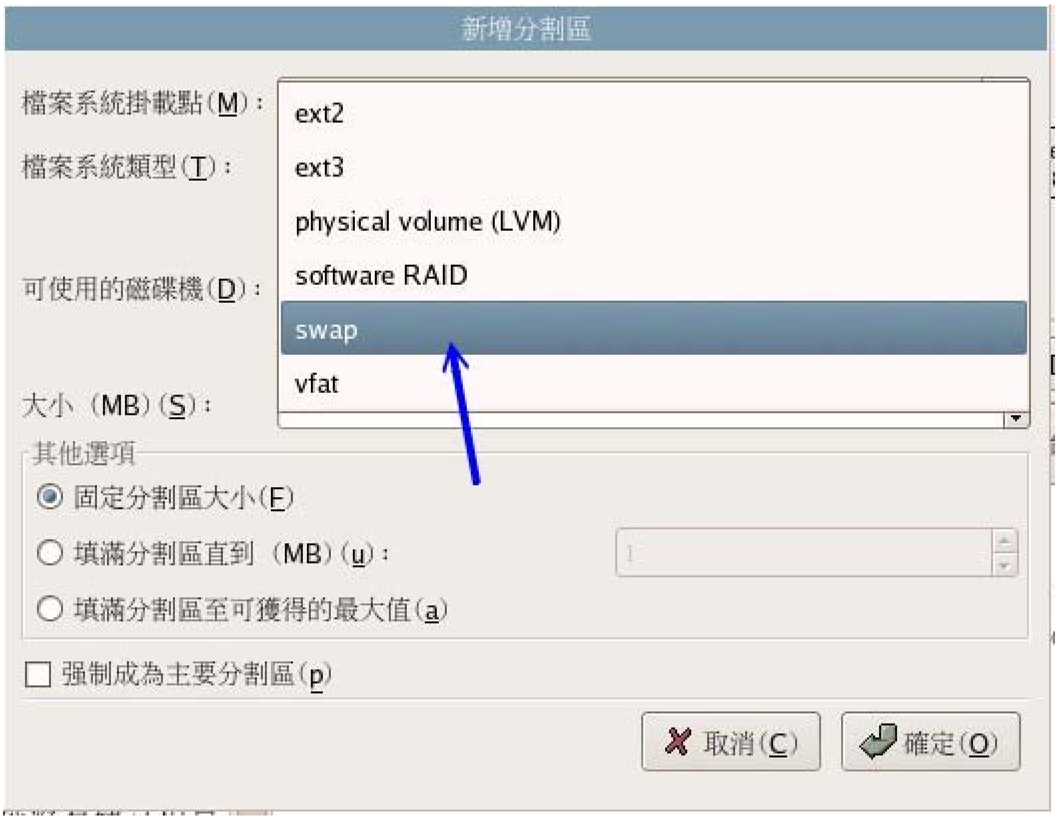


图 18 、新增加 swap 磁盘分割

然后，我们要将 swap 规划为 512 MB ，所以，如下图所示，直接填入 512 ，并且同样的强制为主分割，以及固定分割区的大小，按下确定即可！

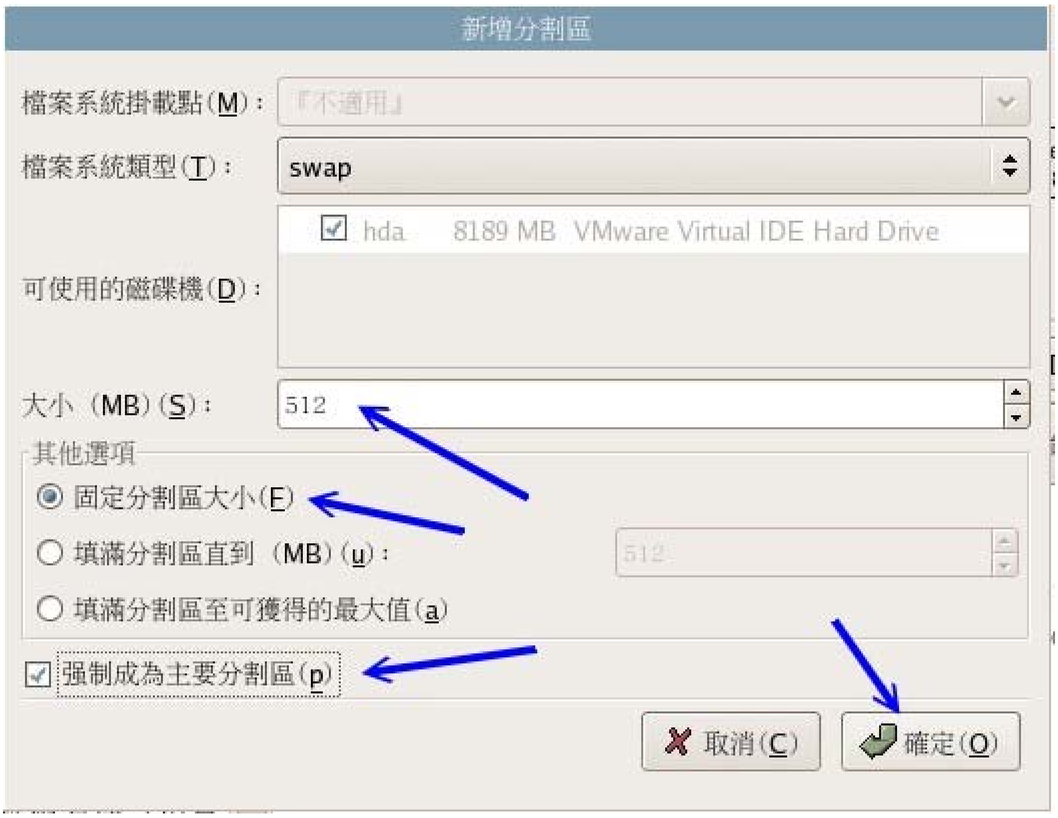


图 19 、新增加 swap 磁盘分割

再次回到主画面！呵呵！看到了吗？又多了一个 partition 出现啦！这次是 /dev/hda2 呢！



图 20 、含有 / 与 swap 的主画面

继续来新增 /home 这个分割槽吧！如下图所示，我们给予 /home 大约 1GB 的磁盘空间吧！ 同样也是选择标准的 ext3 档案系统！

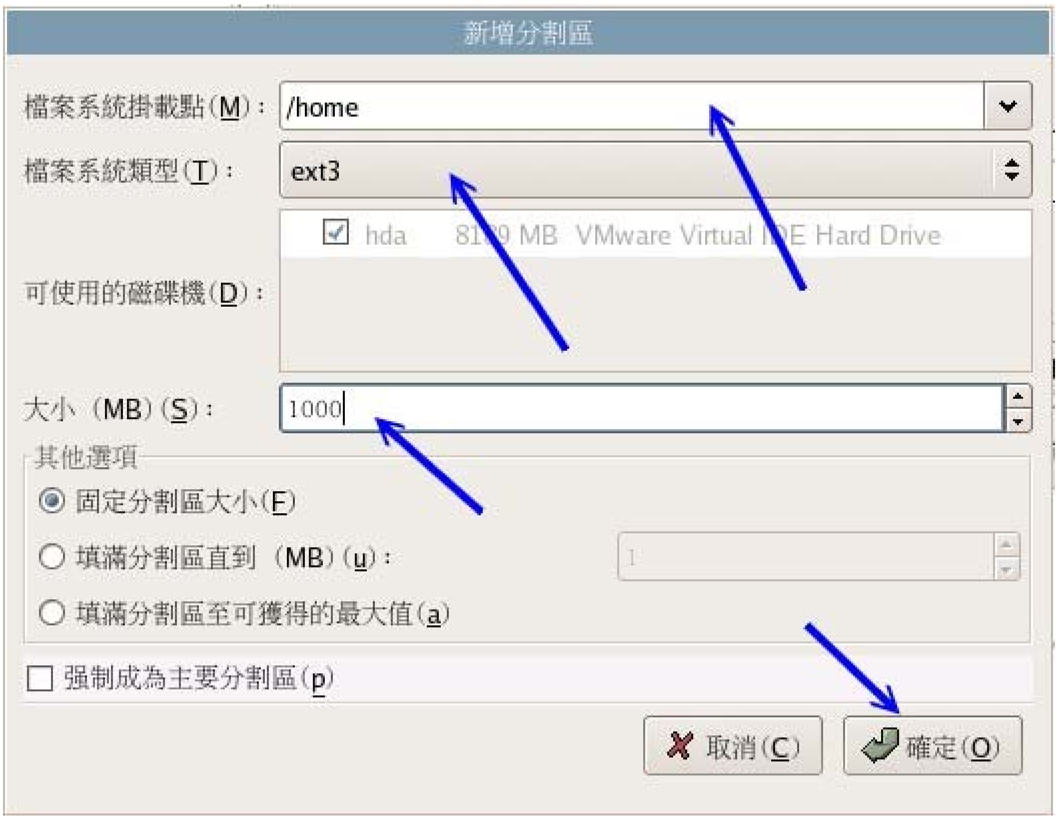


图 21 、规划 /home 的磁盘分割槽

回到主画面后，这就是我们最终的分割结果了！注意到，我们还有一部份的剩余空间没有使用到喔！ 那个没有被使用到的空间，可以做为我们未来的磁盘练习啦！别将他规划了！ ^_^ 另外，您也可以自行测试一下以不同的方式来分割您的磁盘，举例来说，您也可以这样分割：

- Swap 约 100 MB；
- /var 给 3~5 GB；
- /usr 给 3~5 GB；
- / 给 1 GB 以上；
- /home 可以给大一些；
- /backup 用来做为备份的扇区

无论如何，我们这个练习机的分割最终结果如下图：



图 22 、分割完成的最后结果

---

5. 选择开机管理程序：

分割完硬盘之后，接下来就来选择开机管理程序啦！在 Linux 里面主要有 Lilo 与 grub 这两套开机管理程序，不过，目前 Lilo 已经比较少使用，取而代之的就是 grub 这个好用的管理程序啰！ 所以，我们可以看到如下的画面：比较值得注意的是，开机管理程序可以被安装在 MBR 也可以安装在每个 partition 最前面的 super block 处(这个我们会在后面继续说明！看不懂先有印象即可！)。 在下图当中，我们安装在 /dev/hda 内，这就是『MBR』的安装点，如果是类似『/dev/hda1』这个就是 super block 的安装处啦！

在下图中，显示了我们目前仅有一个开机选单，那就是 『Fedora Core』 这个选项而已。 我们可以透过『新增』、『编辑』与『删除』来管理开机时，要显示的项目。 举例来说，如果您有安装 Windows 在这个系统当中，那么此时，您就可以按下『新增』， 将 Windows 可开机扇区加到这个画面当中来喔！至于最底下的『密码』与『进阶设定』， 我们在后续相关的章节谈到时，再深入来研究啊！

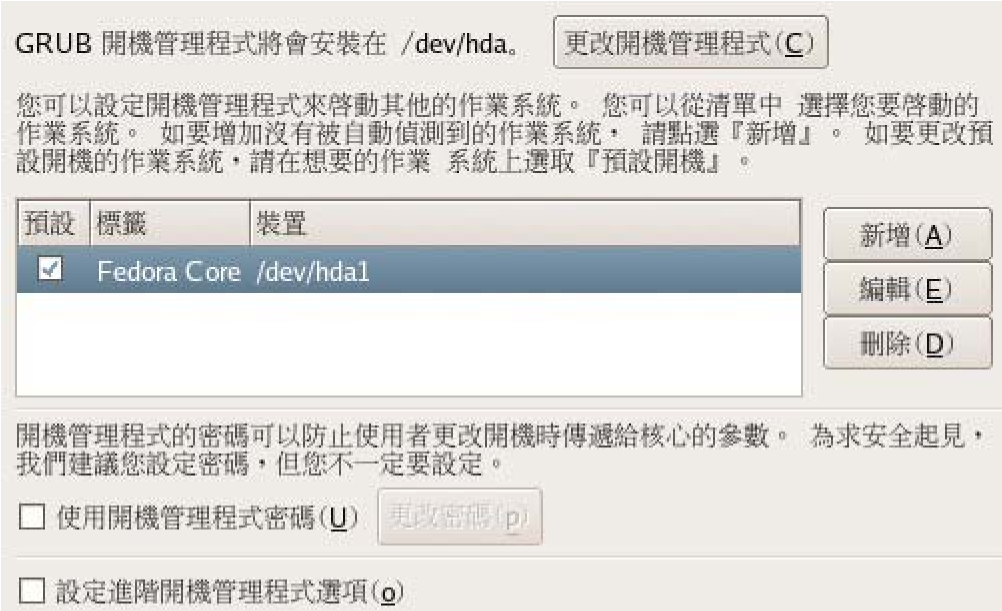


图 23 、安装 grub 开机管理程序

上图中，如果按下『新增』就会出现如下所示的图样，此时，您可以选择所需要的 partition， 以及开机时选单内的名称(标签)，按下确定即可。如果没有额外的开机区， 就略过这个步骤吧！



图 24 、安装 grub 开机管理程序

---

6. 网络与防火墙设定：

再来，如果您的网络卡可以被系统捉到的话，那么您就可以设定网络参数了！ 例如下图所示的模样！目前各大版本几乎都会预设网络卡 IP 的取得方式为『自动取得 IP』， 也就是所谓的『DHCP』网络协议啦！不过，由于这个协议需要有 DHCP 主机的辅助， 开机的过程中可能会等待一段时间。因此，您可以改成手动设定。不过，无论如何， 都要与您的网络环境相同才是。

如果您不懂网络如何设定，没有关系，我们会在服务器篇好好的深入介绍的，在这里，还没有需要了解他！您可以照着鸟哥的设定值设定就好了！未来了解了网络架构，再回来这里进行修订的工作即可！



图 25 、安装程序预设的网络参数设定值

说过啦！不要用 DHCP 啊！利用手动设定即可！你也可以设定开机就驱动网络卡喔！ 如下图的箭头指的地方。至于 IP 嘛！嘿嘿！就跟鸟哥的一样就好了！别担心！



图 26 、设定网络卡 IP

上图中给他按下确定之后，就会出现如下的图示啦！您必须要替自己的 Linux 主机取的名称。 一般来说，我们都不建议取的名称太大众化！因为，可能会造成未来设定主机的一些困扰。 所以，这里鸟哥以自己的名字取一个主机名称呢！您也可以依样画葫芦喔！ 另外，那个网关器与 DNS 的设定嘛！就跟鸟哥设定的一样就好了！不知道原理没关系！ 以后我们再来设定好！



图 27 、设定主机名称与 Gateway, DNS

设定好网络之后，再来则是跟网络有相当大关系的防火墙啦！因为我们是练习用的主机，这里就先不要使用防火墙。 另外，那个 SELinux 的设定值得特别留意！ SELinux 是 Security Enhanced Linux 的简写，这个套件是由 National Security Agency (NAS, http://www.nsa.gov/selinux/) 所发展的，他主要的功能可以代管整个 Linux 系统的存取控制(access control)， 可藉以避免一些可能造成我们 Linux 操作系统安全问题(Security)的软件的破坏， 虽然 SELinux 会有比较好的系统防护能力，不过，如果不熟悉他，那么启动了 SELinux 之后，嘿嘿！ 您未来的服务可能会因为这个较为严密的安全机制，而导致无法提供联机的问题，或者无法进行数据存取的问题， 所以，暂时也将他关闭吧！ ^_^

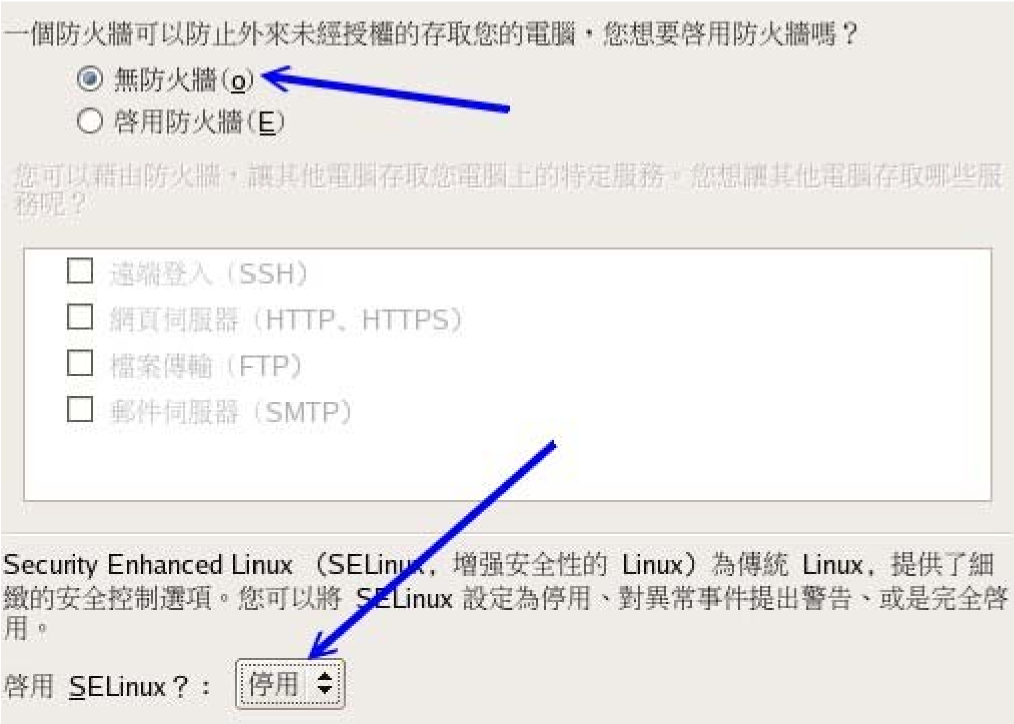


图 28 、防火墙的设定

因为我们停用防火墙，安装程序很好心的会提示我们：『你没有启用防火墙喔！』，嘿嘿！ 没关系！继续吧！因为我们在服务器篇里面会提到自己设定的防火墙功能啊！

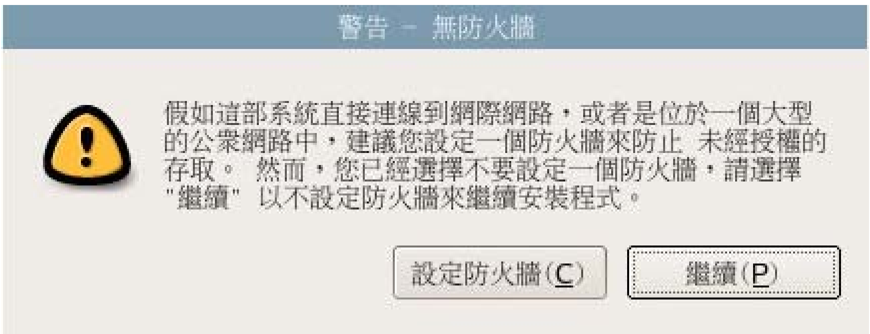


图 29 、无防火墙的警告讯息

---

7. 选择时区与设定 root 密码：

因为全世界被细分为 24 个时区，所以，得要告知系统我们的时区在哪里才行啊！ 如下图所示，您可以选择台北，或直接用鼠标在地图上面点选也可以！ 要特别注意的是那个 『UTC』，他与所谓的『日光节约时间』有关。 不过，我们不需要选择这个，不然的话，还可能造成时区被影响，导致系统显示的时间会与本地时间不同。

Tips:
事实上， UTC 与所谓的 GMT 时间是一样的！就是格林威治时间，那是标准的地球时间啦！ 以格林威治(英国)所在地为 GMT 0 点，而将地球切为 24 个时区,我们台湾在 GMT 的东方，时间比较早，所以台湾本地时间为 GMT+8 小时。





图 30 、时区的选择

再来则是最重要的『系统管理员的密码』设定啦！ 在 Linux 底下，系统管理员的预设名称为 root ，请注意，这个密码很重要！ 虽然我们是练习用的主机，不过，还是请您养成良好的习惯，最好 root 的密码可以设定的严格一点。 可以设定至少 8 个字符以上，而且含有特殊符号更好，例如：I&my_dog 之类，有点怪， 对您又挺好记的密码！

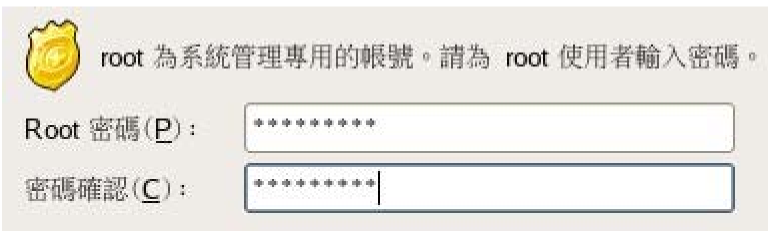


图 31 、系统管理员密码的设定



图 32 、系统开始读取套件数据

---

8. 套件的选择：

在进行完套件的读取之后，接下来，则是选择您要的套件啦！ 咦！我怎么知道我要什么套件？哈哈！您当然不可能会知道～知道的话.....就不会来这儿查阅数据了 @_@ 没有啦！开开玩笑....呼～好冷～～

基本上，鸟哥不建议您使用安装程序预设的套件来安装！因为，会缺乏很多需要的套件的！ 如果您的硬盘够大，建议您，像下图一样，选择『全部安装』，一劳永逸！不必怕什么咚咚没有装。 当然啦，这是针对练习机来进行的安装。

如果您已经具有基本的套件管理知识，那么鸟哥会建议您选择『最小值』来安装， 不过，要有心理准备，就是很多数据您都得在安装成功后，再自行由光盘中的档案来安装！ 但是，优点是，会占用比较少的空间，而且系统会比较干净。

那有没有折衷的方法啊？有的，假设您不需要 X Window ，但却需要一些有的没有的工具的话， 那么您可以选择底下这些相关的套件啊！

- 编辑器
- 文字接口的因特网
- 编写与出版
- 服务器设定工具
- 开发工具(这个最重要！一定要选择！)
- 兼容旧式软件开发
- 语言支持
- 管理工具
- 系统工具
- 打印支持

这样的套件大约需要 1833 MB 的硬盘空间。而如果您想要使用 X Window 的话，那么最上方的 『X 窗口系统』与 『KDE 桌面环境』也可以将他选择的啦！仔细注意到下图的最下方， 有个『总安装大小： xxxxxM』吧？！ 那就是您选择的套件总共会占用多少硬盘空间啦！ 选择看看吧！



图 33 、套件的选择

检查过相依属性的问题后，会出现一个说明窗口，里面说的是， 等一下就会主动的将所有选择的套件安装妥当，而且，所有安装的信息都会被纪录在 /root/install.log 及 /root/anaconda-ks.cfg 这两个档案中呢！



图 34 、安装前的说明

由于您所选择的套件分别在这几张光盘当中，所以会出现这个窗口，告知您， 必须要有这四片光盘才可以安装妥当喔！

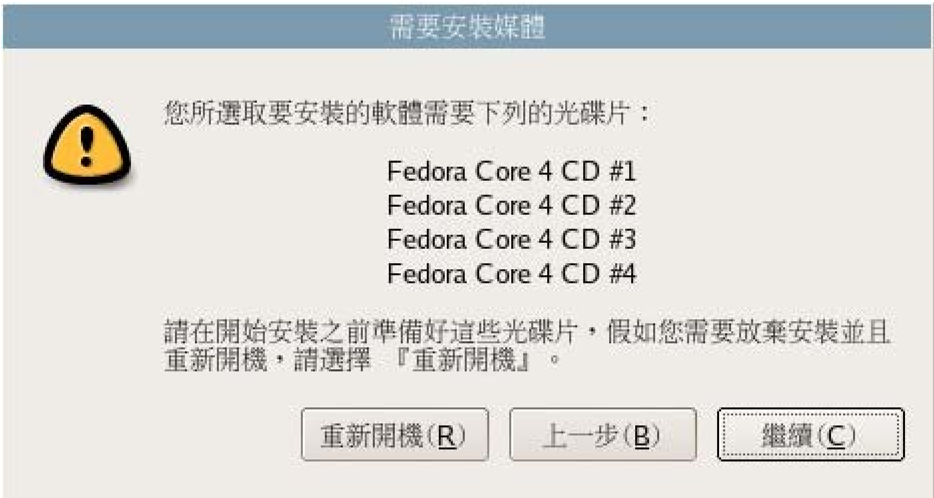


图 35 、确定您所拥有的光盘

呵呵！开始安装啰！在安装的这个画面中，会显示还需要多少时间， 每个套件的名称，以及该套件的简易说明呢！



图 36 、安装过程的画面

一片一片的将光盘拿出来～放进去～花费的时间可不少呢！



图 37 、光盘更换警示

等到所有的安装光盘都安装之后，一切就都 OK 了！最后出现这个画面， 请将光驱的片子拿出来，准备按下『重新开机』去开机吧！



图 38 、光盘更换警示

在重新开机时，会出现如下的画面，这是正常的！别担心！系统就要重新开机啰！ 接着请看下一章相关的开机与关机内容啊！

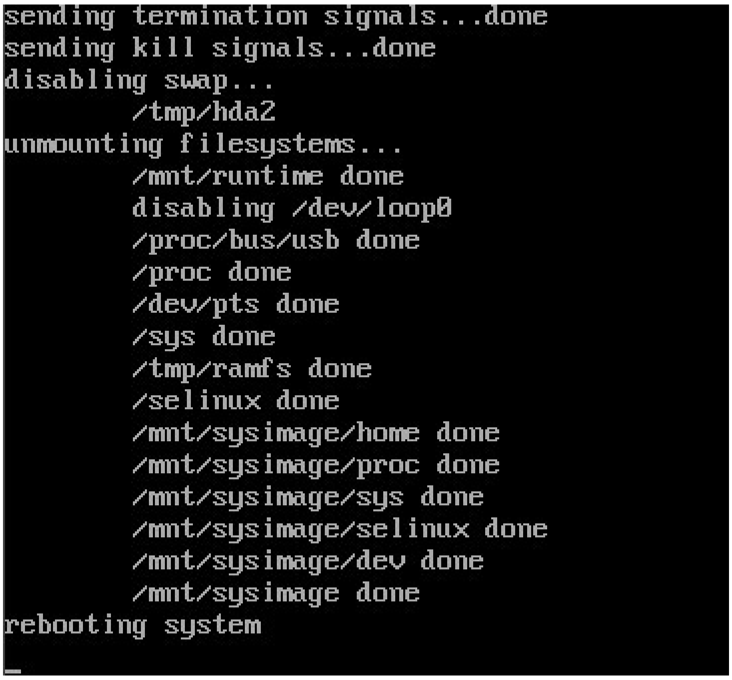


图 39 、光盘更换警示

---

9. 其它注意事项：

在安装完毕之后，有些地方还是需要提醒您的：

- 你可能会觉得很奇怪，为什么你的安装过程会跟我的不一样？！ 呵呵！因为每个人选择的套件都不尽相同，因此， 如果你在安装的过程中选择了跟我不一样的套件，不用担心，安装过程会有些许的不相同是正常的！
- 在安装完成之后，请千万记得『取出光盘片』，不然又会在进入一次安装画面喔！
- 同时建议，安装完成之后，请进入您的 BIOS 当中，将开机的顺序改回来『 C、A 』或『 C only 』反正就是让硬盘开机啦！这样比较安全一些！

好了！这样应该就已经安装完毕了！请继续往下看看吧！而且，相当的建议您，在正式的进行架站之前， 请依序看一下底下的网页，最好不要跳着看，不然的话，嘿嘿嘿嘿！出现什么问题可不要怪我！因为，照着顺序看会对你的 Linux 认识比较有帮助啦！

附带额外提醒一点，由一些 bug reports 的数据显示， FC4 预设的 X Window 对于 Intel 以及一些 G550 的显示卡在支持度上可能有问题，这是因为编译的过程里面下达的参数不佳所致。完整的 bug reports 可以参考：

https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=161242

简单的来说，就是 Intel 的显示卡与 G550 的显示卡硬件在 FC4 的编译过程中， 可能无法对这两种显示卡做比较好的处理，导致 tty1 ~ tty7 都没有东西跑出来。 那怎么解决呢？如果您是初次接触 Linux 的话，那么请先参考 Linux 档案与目录管理 章节当中的 cp 这个复制指令的介绍，然后将 FC3 的这个档案 /usr/X11R6/lib/modules/libvgahw.a 复制到 FC4 底下的相同档案去， 经过重新开机后，应该可以克服这个问题。该档案可以在底下取得：

ftp://people.redhat.com/mharris/libvgahw.a

在这里或许您还看不懂上面写些什么，没关系，这很正常，等到下一章后， 您就晓得这个地方在讲啥了～ 所以，如果您在下一章的文字、图形接口变换时发生问题， 记得来这里看看呦！

多重开机安装流程与技巧

很多的朋友，包括我自己，由于工作的需要，常常需要两部不同的操作系统来处理日常生活与工作的杂事！ 那么我是否需要两部计算机呢？并不需要，只要一部计算机使用多重开机的方式来进行安装，嘿嘿！这样就 OK 啊！理论上是如此，不过实际上还需要一些小技巧呢！

不过，就如同鸟哥之前提过的，多重开机系统是有很多风险存在的，而且您也不能随时变动这个多重操作系统的开机扇区， 这对于初学者想要『很猛烈的』玩 Linux 有点妨碍～所以，鸟哥不是很建议新手使用多重开机啦！ 所以，底下仅是提出一个大概，您可以看一看，未来我们谈到后面的章节时， 您自然就会有『豁然开朗』的笑容出现了！ ^_^

- 硬盘重新规划的多重开机系统：

如果你想要在你的 Linux 机器上同时安装 Windows ？可行吗？当然可行啰！况且目前很多的朋友手边只有一部计算机，但是又想要同时学习一下 Linux ，呵呵！那么安装多重操作系统实在是必须要的！好了！那要如何安装呢？以鸟哥前一阵子帮一个朋友规划的 Win98, Win2000, Linux 为例，我先将硬盘以 spfdisk 切割成两个 FAT partition，分别是 2GB 与 3GB ，预计安装 Win98 与 Win2000 （分别是 C: 与 D: ），然后再以 CD 开机后，分割最后的磁盘成为 / 与 Swap 两个！好了！如何安装：

1. 先以 Spfdisk 分割硬盘：
   由于 Windows 的 Fdisk 实在太慢了，我蛮喜欢使用 spfdisk 这个全中文的磁盘分割接口的！简单又方便！将硬盘切割成 C: 2GB, D: 3GB即可！详细的 Spfdisk 执行范例可以到网络上搜寻一下教学文章吧！ 例如：http://linux.vbird.org/linux_basic/0140spfdisk.php

2. 先安装 Win98 ：
   这个简单吧！用 98 开机片开机之后，直接安装，并且选择安装在 C 槽即可！

3. 再安装 Win2000：
   进入 Win98 之后，将 Win2000 的光盘片放进光驱中，屏幕会自动的跑出一个窗口， 问你要不要升级，选择『是』，然后会进行一些小动作！在安装程序问到 『升级安装或全新安装』的时候，请千万选择『全新安装』这个项目， 并且不要升级硬盘扇区！然后在出现一个『问你安装目录所在』的问题时， 进入选项里面，选择『要我自己挑选硬盘分割区』那个项目！然后接下来一直按下 『确定』或『是』即可！之后，计算机会重新开机，开机完成之后会进入 Win2000 的安装画面，然后在出现『安装扇区』的时候，请选择 D 槽， 并且选择『不要更改扇区档案系统』即可！接下来就会完成一些程序啦！

4. 最后才安装 Linux distribution ：
   是的，最后才安装 Linux ！安装的过程就如同上面提的， 只不过在硬盘分割的地方会比较不一样就是了！！

5. 以 Lilo 或 grub 设定多重开机：
   是的，您必须选用 lilo 或 grub 来将您的开机程序设定一下， 这个动作我们会在后头再谈，或者您可以在了解 vi 之后，直接翻到多重开机章节去瞧一瞧去！

* 在既存的 Windows 系统中加装 Linux 系统：

另外再提供一个之前也曾经安装过的一个经验！恩！你可能会觉得奇怪， 这个方法跟上一个方法有什么不同！？呵呵呵呵！最大的不同在于：

我既存的 Windows 系统中的数据不想丢掉， 并且我也没有新的硬盘来暂存我的系统或者是备份数据！假设原本我的 20 GB 硬盘中分割成 10GB, 10GB 两槽，但是我还想要安装 Linux ， 且是在『旧系统仍然可以存活』的情况下！那该如何是好？！

这真的是很有趣的问题！早先在 Windows 系统中， 鸟哥就犯了一个错！C 槽给的太大了！基本上，系统文件不需要太大啦！通常我都喜欢 C 槽只给大约 4 GB 左右的空间（甚至更小），这是因为 C 槽是很需要备份的！如果太大的话， 备份很麻烦！所以系统重置就会很花时间（因为所有的东西都要重新安装！我哩咧....！）！因此，我都习惯将 C 槽只给一点点的空间，然后再安装完并设定完所有的系统之后，马上以 Ghost 来备份我的系统！而所有的备份数据文件都摆放在 D 槽！此外，我的 OutLook Express 的书信目录也都不是摆在 C 槽！呵呵所以我不会很害怕 C 槽挂掉，因为，直接以 Ghost 还原即可啰！系统还原还不需要 30 分钟呢！

这里就发生一个问题啦，假如原本的系统是 10GB, 10GB 的两槽，不过全部的有用到的资料量只有 10GB 不到！也就是还有空间来安装 Linux ，但是由于硬盘切割的不好，所以伤脑筋！此外， 我的原系统希望留下来，而且也希望可以安装 Linux ，要怎办？！鸟哥曾经这样做过：

* 由于 FAT 的扇区使用，其实只是在磁头区域（所谓的硬盘第零轨）规划而已， 所以，我就将我的数据先以『磁盘重组』的方式将数据都归结在一起；
* 然后以 Spfdisk 将该硬盘的 FAT 表进行分割，注意喔！只是分割 FAT 表，并没有 format 喔！不过这里的技术性很高，需要特别注意！因为你是将 FAT 表重新划分，所以你的数据必须要在同一个扇区内！好了，我就将原本的 10GB 10GB 切割成 4GB、10GB 与 6GB 三槽！而且在 spfdisk 的帮助之下， 顺利的在没有任何数据遗失的状况下，将我的硬盘由原先的两槽分割成三槽啰！那么一来， 我就可以在我原本的 D 槽里面安装 Linux 啦！方法有点像底下的图示：

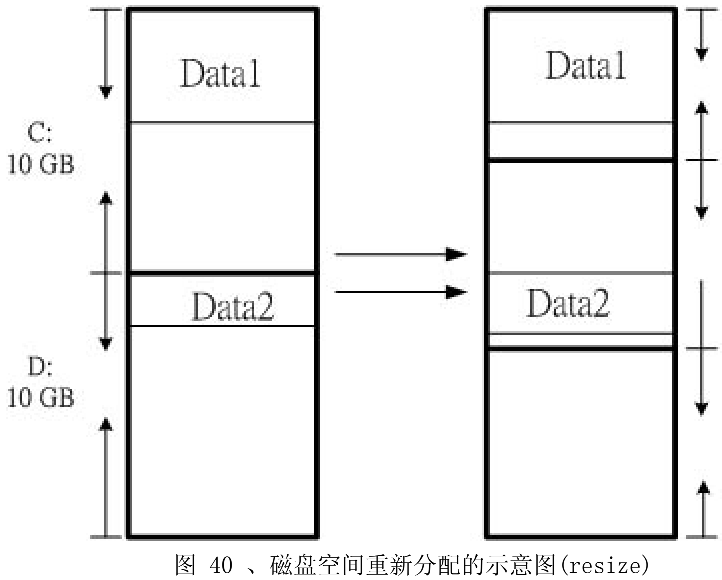


图 40 、磁盘空间重新分配的示意图(resize)

很神奇吧！数据还是在原来的地方，不过扇区的定位点改变了，还多出一个扇区！ 不过，这里要提醒大家，虽然鸟哥曾经以这个方法成功的将硬盘数据在不毁损的情况下， 顺利的将硬盘切割完毕，不过，这个方法

本身还是具有相当程度的风险，呵呵！ 因此不是很建议您这样做！尤其是当你的数据还很重要的时候！切记切记！

- 在既存的 Linux 系统安装新的 Linux 系统：

那我能否在 Linux 系统上面安装另一套 Linux 呢？举例来说，在已经安装了 SuSE 的 Linux 上面加装 Fedora 4 呢？当然可以啊！不过，同样的，您依然有着 partition 分割的问题。 如果要将原本的 partion 在 Linux 里面放大或缩小时，您必须要了解整个档案系统的概念， 这点要参考后续章节的磁盘与档案系统的相关知识， 然后再利用基础篇最后几章的硬件维护与管理里面谈到的 resize2fs 等指令，就可以达到啰～不过，同样的，也是很危险喔！ 没有三两三，不要上梁山！对吧～先不要急着玩这么进阶的玩意儿啊！ ^_^

关于大硬盘导致无法开机的问题

有些朋友可能在第一次安装完 Linux 后，却发现无法开机的问题， 也就是说，确实可以使用上面鸟哥介绍的方法来安装 FC4 ，但是，就是无法顺利开机， 只要重新开机，就会出现类似底下的画面：

```
# 前面是一些奇怪的提示字符啊！
grub> _
```

然后等待你输入一些数据～如果不幸你发生了这样的问题，那么可能的主要原因就是.....

- 你的主机板 BIOS 太旧，导致捉不到您的新硬盘；
- 你的硬盘容量太大了 (例如超过 120 GB 以上)，但是主机板并不支持～

如果真的是这样，那就麻烦了～你可能可以这样做：

- 前往您主机板的官方网站，下载最新的 BIOS 档案，并且更新 BIOS 吧！
- 将您的硬盘的 cylinders, heads, sectors 抄下来，进入 BIOS 内，将硬盘的型号以使用者设定的方式手动设定好～

当然还有一个最简单的解决方法，那就是：

> 重新安装 Linux ，并且在 partition 的地方，建立一个 100MB 左右的 partition ，将他挂载到 /boot 这个挂载点。并且要注意， /boot 的那个挂载点，必须要在整个硬盘的最前面！例如，必须是 /dev/hda1 才行！

至于会产生这个问题的原因确实是与 BIOS 支持的硬盘容量有关， 处理方法虽然比较麻烦，不过，也只能这样做了。更多与硬盘及开机有关的问题， 鸟哥会在 开机与关机程序 再进一步说明的啦！

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- Linux 的目录配置以『树状目录』来配置，至于磁盘分割区 (partition) 则需要与树状目录相配合！请问，在预设的情况下，在安装的时候系统会要求你一定要分割出来的两个 Partition 为何？

  就是根目录『/』与虚拟内存『Swap』

- 什么是 IDE 界面，一般而言，普通 PC 允许几个 IDE 界面与装置?

  IDE 为用来传输硬盘数据的一个汇流界面；共有 IDE1, IDE2 ，分别有 master 与 slave 所以共四个 IDE 装置支持!

- IDE2 的 master 之第一个 logical 磁盘中，其装置代号（文件名称）为何?

  /dev/hdc5

- 在硬盘分割 (Partition)时，最多有几个 primary + extended ?

  Primary + Extended 共四个，其中 Extended 只有一个！ 更详细的硬盘与 MBR 可以参考 http://phorum.vbird.org/viewtopic.php?t=182

- 若在分割的时候，在 IDE1 的 slave 硬盘中，分割『六个有用』的扇区（具有 filesystem 的），此外，有两个 primary 的扇区！请问六个扇区的代号?

  /dev/hdb1(primary)
  /dev/hdb2(primary)
  /dev/hdb3(extended)
  /dev/hda5(logical 底下皆为 logical)
  /dev/hda6
  /dev/hda7
  /dev/hda8
  请注意，5-8 这四个 logical 相加的总和为 3!

- 一般而言，在 RAM 为 64MB 或 128 MB 的系统中， swap 要开多大?

  Swap 可以简单的想成是虚拟内存，通常他的建议大小为 RAM 的两倍， 但是实际上还是得视您的主机规格配备与用途而定。约两倍的 RAM ，亦即为 128 MB 或 256 MB ，可获得较佳效能!

- 什么是 GMT 时间？台北时间差几个钟头?

  GMT 时间指的是格林威治时间，为标准的时间，而台北时间较 GMT 快了 8 小时!

- Tap, SCSI 硬盘, RAID, printer 的装置代号?

  Tape: /dev/ht0 (IDE), /dev/st0 (SCSI);
  SCSI H.D. : /dev/sd[a-p];
  RAID : /dev/md[0-15];
  printer: /dev/lp[0-2]

- 如果我的磁盘分割时，设定了四个 Primary 扇区，但是磁盘还有空间，请问我还能不能使用这些空间?

不行！因为最多只有四个 Primary 的磁盘分割槽，没有多的可以进行分割了！且由于没有 Extended ，所以自然不能再使用 Logical 分割说

- 通常在安装 Linux 的时候，最重要的就是磁盘分割了！请问：磁盘分割通常要分成几个步骤？

  1. 进行磁盘分割 partition ；
  2. 进行格式化 format ；

- 磁盘分割之后会有所谓的 Primary, Extended 与 Logical 的磁盘分割槽，请问何者为可使用的 Partition ？

  只有 Primary 与 Logical 为可用， Extended 为不可直接使用的 Partition ， 还需要再次的分割成为 Logical 之后，才可以继续使用！而最大可分割出来的 Partition 应该有 63 个才对！

- 硬盘最小的物理储存量(sector)大小通常为多少？

  目前个人计算机的 SATA/IDE 接口硬盘 sector 的单位为 512 bytes。

- 硬盘的第零轨含有 MBR 及 partition table，请问，partition 的最小单位为(磁柱、磁头、磁道)

  为 Cylinder (磁柱)，所以 partition 的大小为磁柱大小的倍数。

安装 Mandrake 9.0 与多重开机技巧

最近更新日期：2003/02/05

Linux 安装的第一步『规划』

由第二章的内容我们大致上可以了解：『Linux 主机的配备要求与 Linux 主机所提供的服务有关』，所以啰，要安装 Linux 的第一步就是先进行 Linux 主机的未来蓝图规划啦！那么该如何规划？由第二章的内容，我们不难了解，要规划好一个 Linux 主机的话，需要：

1. 决定 Linux 主机的角色定位；
2. 根据步骤一，选择适当的硬件配备；
3. 根据步骤一，决定实体硬盘的分割状态；
4. 根据主机上面的操作系统，选择适当的开机加载程序 (boot loader)，以便安装在主要开机扇区中 (Master Boot Recorder, MBR)；
5. 根据步骤一，选择所需要的 Linux 程序套件；

好了，既然已经知道大概需要规划的原理之后，底下我们就实地的来操练一遍吧！好让您不会很快的忘掉去！

> 假设：我是我们宿舍的代表，由于同住的校外宿舍同学需要上网缴交作业，但是该宿舍只有一条电话线，因此希望以 ADSL 来做为网络联机的方式。也就是说，我们宿舍里面要安装一部 Linux 主机来做为频宽分享的机器，同时，这部 Linux 主机预计也要做为我们宿舍里面 20 个人的邮件主机，与网页空间服务器，请问我该如何规划我的 Linux 主机呢？

1. Linux 主机的角色定位：
   由上面的说明，可以知道 Linux 主机的服务主要有 NAT (Network Address Transfer) 这个频宽分享的机制、邮件服务与 Web 空间提供等等，此外，为了以后方便系统升级与安装其它套件，因此需要安装一些工具软件，例如 gcc 这个编译器与 kernel-header 或 kernel-source 等等套件。

2. 硬件的配备选择：
   由于服务的对象并不多，加上主机的服务当中，需要 CPU 运算的地方较少，不过呢，由于我需要提供每个人的磁盘使用空间，并且还要提供使用者的邮件空间，所以硬盘方面可能需要大一点容量才行。所以我的硬件配备可以是：

   - 主机板与 CPU ：CPU 只要比 P-166 好即可，而主机板需要与 CPU 形式配合。此外，选择淘汰的计算机配备来安装就很好了；
   - 内存：使用 64 MB 以上的 RAM ，如果未来网页空间的流量太大时，可能需要提升内存到 256 MB 以上，所以需要预留内存插槽；
   - 硬盘：硬盘至少需要 3.2 GB 以上的 IDE 硬盘；

- 网络卡：网络卡预计使用螃蟹卡；
- 显示卡：由于这部机器本身是做为主机之用，所以不需要 X-Window ，因此显示卡使用的是 S3 Virge 的 PCI 显示卡；
- 安装过程中需要的装置：键盘、屏幕、光驱、软盘机等等，这些装置在安装完成 Linux 之后，即可马上拔掉！

3. 关于硬盘分割的分配：
   由于我们的 Linux 主机要用做网页空间与邮件主机，所以如果为了安全起见，最好将放置网页的目录与放置邮件的目录安置在不同的扇区中，因此总共规划为四个扇区，分别为：

   - /
   - /var/spool/mail
   - /home
   - swap

   特别注意到，硬盘分割的分配与你的主机规划相关性最高了，在下一节当中，我们会更仔细的介绍硬盘分割的基本原理与步骤！而至于各个目录与扇区的相对应关系，我们将在磁盘档案系统进行说明，请莫着急呦 ^_^ ！

4. 选择 MBR 当中的开机管理程序：
   在 Linux 里面预设使用两种开机管理程序，分别是 LILO 与 GRUB 这两个好东西，其中，LILO 算是比较早期的开机管理程序，不过， VBird 个人倒是还蛮喜欢 LILO 的，毕竟他虽然比较死板，但是还蛮好用的，磁盘代号设定上面与 Linux 上的磁盘代号相同，所以没有什么太大的困扰。而较新的 GRUB 其实是很棒的一套开机管理程序，我个人认为，他最大的功能也最具魅力的地方是具有『动态搜寻核心档案』的功能，他可以让您在开机的时候，可以自行编辑您的开机设定系统档案，呵呵！所以即使您不小心设定错了 grub ，没关系！开机的时候自行编辑一下就好啦！这方面的技巧，我们会在开机流程与 Loader 的时候再来详细的介绍，还是慢慢的从头学习起来呦！

5. 选择所需套件：
   由于将光盘上面的全部套件都安装，是有点浪费硬盘空间，当然如果您是要练习 Linux 的话，那么还是完整的都给他安装下去的好。由于我们需要 Web 与邮件，所以需要特别加选这两个套件来安装，此外，由于预设的安装项目并不包含 gcc,kernel-headers 等对于自行编译程序者而言相当重要的套件，所以我们也要额外加选这些项目！

到了这一步之后，嘿嘿！规划就已经差不多了，所以，这个时候，基本上已经可以开始来安装 Linux 啦！但是，还是有个困扰耶，那就是，在第三步骤的时候，我要怎么在安装的时候分割我的硬盘呀！？第二章里面有提过硬盘的排线与硬盘在 Linux 里面的磁盘代号有关，那么该如何分割？另外，有什么自订的方式可以来帮我分割硬盘吗？呵呵！底下我们就来提一提如何

硬盘分割之配置

硬盘分割与配置的好坏，会影响到未来您的主机的使用情况，此外，好一点的分割方式，会让您的数据保有一定的安全性！怎么说呢？这么想好了，如果你的 Windows 硬盘里面，仅有 C 槽的话，那么当 Windows 需要重新安装的时候，你又想要重新格式化 （ format ） 时，而 C 槽里面很不巧的，已经放了很多重要的档案数据，这个时候怎么办？光是搬这些重要数据到其它空间就受不了！所以，比较聪明的玩家，都喜欢分割成两槽以上，将系统档案与数据文件分开，可以达到比较好的管理效果！

所以啰，正常使用情况下的 Linux 主机，通常会依照目录与主机的特性，来分割硬盘，以达到比较好的管理成效。不过，由于 Linux 的硬盘分割比较具有弹性，同时， Linux 硬盘分割程序 fdisk 功能很强悍，此外，要分割的好，必须要了解一下基础的硬盘架构，所以，底下我们先来介绍一下硬盘的基本架构，然后再来介绍如何分割吧！

- 硬盘连接排线与硬盘代号：

  通常在 586 之后生产的主机板上面都有两条接排线的界面 （ 排线就是用来连接硬盘与主机板的那一个东西啦！ ），而我们称这种界面为 IDE 界面，这也是目前的主流硬盘界面，为了区隔硬盘读取的先后顺序，所以主机板上面的这两个界面就分别被称为 Primary （ 主要的 ） 与 Secondary （ 次要的 ） IDE 接口啰，或者被称为 IDE1 （ Primary ） 与 IDE2 （ Secondary ）。而如果你有仔细观察的话，那么每一条排线上面还有两个插孔，也就是说一条排线可以接两个 IDE 界面的装置 （ 硬盘或光驱 ），而你有两条排线，因此一个主机板在预设的情况中，应该都可以接四个 IDE 界面的装置。好了，那么每条排线上面该如何判别哪一个是主硬盘 （ Master ），哪一个是副硬盘 （ Slave ） 呢？这个时候就需要调整硬盘上面的跳针 （ jump ） 才可以知道！请察看一下您的硬盘机吧！上面应该都会有图示说明才对。

  好了，所以如果我有一个光驱了，那么我最多就只能再安装三部 IDE 接口的硬盘在我的主机上面。OK！那么由于我的硬盘与 Linux 的磁盘代号有关，那么我怎么知道这个硬盘的代号呢？没问题啦，由 IDE 1 （ Primary IDE ） 的 Master 硬盘先计算，最后是 IDE 2 的 slave 硬盘，所以各个磁盘的代号是：

IDE\Jumper	Master	Slave
IDE1(Primary)	/dev/hda	/dev/hdb
IDE2(Secondary)	/dev/hdc	/dev/hdd

  假如我只有一颗硬盘，而且这一颗硬盘接在 IDE 2 的 Master 上面，那么他在 Linux 里面的代号就是 /dev/hdc 啰！OK！好像没问题了呦！呵呵！才不是呢，问题很大呦！因

为，如果我这个磁盘被分割成两槽，那么每一槽在 Linux 里面的代号又是如何？注意！基本上，在 Linux 底下我们不是用 槽 为单位，而是以 partition (磁盘分割区块) 来说明！所以啰，如何知道每个 partition 的代号呢？

- 认识硬盘：

  基本上，硬盘是由最小的组成单位 扇区 (sector) 所组成的，而数个扇区组成一个磁柱 (cylinder) ，最后构成整个硬盘的容量大小。关于硬盘的管理我们在后续章节再来介绍，这里我们比较想要知道的是，如何分割硬盘，所以先简单的将硬盘变成如下的图标：



在上面的图示中，我们可以很清楚的知道，在硬盘里面有分为两个区域，一个是放置这个硬盘的信息区，我们称为 Master Boot Recorder, MBR (主要开机扇区)，一个则是实际档案数据放置的地方。MBR 可以说是整个硬盘最重要的地方了，因为在 MBR 里面记录了两个重要的东西，分别是：开机管理程序，与磁盘分割表 (partition table)。因此，只要 MBR 物理实体坏掉了，那么这颗硬盘就差不多要报废了！因为，如果系统找不到 partition table ，就无法使用这块硬盘，所以数据即使没有丢掉，但是没有 MBR ，呵呵，还是不能使用的啦！

首先来看一看什么是 partition table 呢？简单的说，我们说的『硬盘分割』就是在修改这个 partition table 而已！他基本上定义了『第 n 个磁盘区块是由第 x 磁柱到第 y 个磁柱』，所以，每次当系统要去读取 n 磁盘区块时，就只会去读取第 x 到 y 个扇区之间的数据！呵呵！这样知道了吗？很简单吧！下次记得人家在谈磁盘分割的时候，不要以为系统真的会在硬盘上面用力、努力的划标签！实际上，他最大的功能就是修改 MBR 里面的 partition table 啦！不过，由于这个 MBR 区块的容量有限，所以，当初设计的时候，就只有设计成 4 个分割纪录，这些分割记录就被称为 Primary (主分割) 及 Extended (延伸分割) ，也就是说，一颗硬盘最多可以有 4 个 Primary + Extended 的扇区，其中， Extended 只能有一个，因此，你如果要分割成四块磁盘分割的话，那么最多就是可以：

P ＋ P ＋ P ＋ P
P ＋ P ＋ P ＋ E

的情况来分割了。其中需要特别留意的是，如果上面的情况中， 3P +E 只有三个『可用』的磁盘，如果要四个都『可用』，就得分割成 4P 了！(因为 Extended 不能直接被使用，还需要分割成 Logical 才行，底下我们会继续说明的！)。那么为什么要有 Extended 呢？这是因为如果我们要将硬盘分割成 5 的磁块的话，那么怎么办？这个时候就需要 Extended 的帮忙了，本身 Extended 是不能在任何系统上面被使用的，还需要再额外的将 Extended 分割成 Logical (逻辑) 分割才能被使用，所以啰，藉由这个 Extended

的帮忙，我们就可以分割超过 5 个可以利用的 partition 啰！不过，在实际的分割时，还是容易出现问题的，底下我们来思考看看：

- 思考一：如果我要将我的大硬盘『暂时』分割成四个 partition ，同时，还有其它的空间可以让我在未来的时候进行规划，那么该如何分割？
  说明：

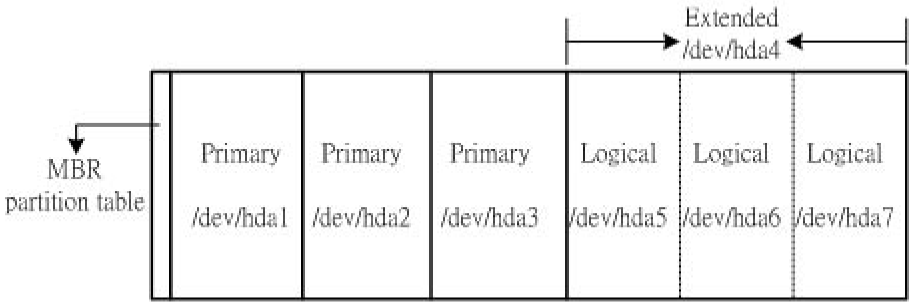
  由刚刚的说明，我们可以知道， Primary + Extended 最多只能有四个 partition，而如果要超过 5 个 partition 的话，那么就需要 Extended 的帮忙。因此，在这个例子中，我们『千万不能分割成四个 Primary 』为什么呢？假如您是一个 20 GB 的硬盘，而 4 个 primary 共用去了 15 GB ，您心想还有 5 GB 可以利用对吧？错！剩下的 5 GB 『完全不能使用』，这是因为已经没有多余的 partition table 纪录区可以记录了，因此也就无法进行额外的分割，当然啰，空间也就被浪费掉了！因此，请千万注意，如果您要分割超过 4 槽以上时，请记得一定要有 Extended 分割区，而且必须将所有剩下的空间都分配给 Extended ，然后再以 logical 的分割区来规划 Extended 的空间。

- 思考二：我可不可以仅分割 1 个 Primary 与 1 个 Extended 呢？
  说明：

  当然可以！基本上， Logical 可以有 64 个，因此，你可以仅分割一个主分割，并且将所有其它的分割都给 Extended ，利用 Logical 分割来进行其它的 partition 规划即可！

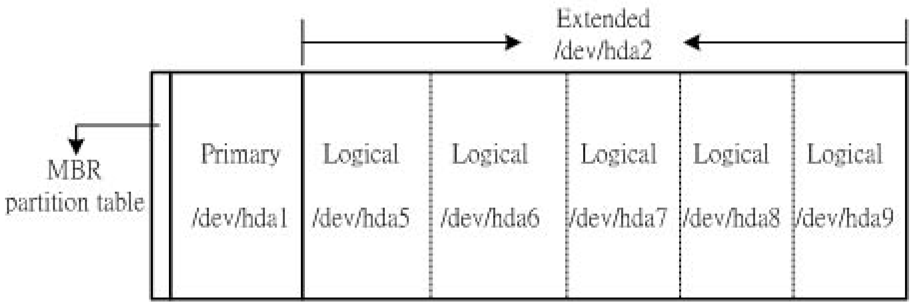
- 思考三：假如我的硬盘安装在 IDE 1 的 Master ，并且我想要分割成 6 个可以使用的硬盘扇区，那么每个磁盘在 Linux 底下的代号为何？
  说明：

  由于硬盘在 Primary ＋ Extended 最多可以有四个，因此，在 Linux 底下，已经将 partition table 1 ~ 4 先留下来了，如果只用了 2 个 P + E 的话，那么将会空出两个 partition number 呦！再详细的说明一下，假设我将四个 P + E 都用完了，那么硬盘的实际分割会如同下图所示：



实际可以使用的是 /dev/hda1, /dev/hda2, /dev/hda3, /dev/hda5, /dev/hda6, /dev/hda7 这六个 partition！至于 /dev/hda4 这个 Extended 扇区本身仅是用来规划出让 Logical 可以利用的磁盘空间而已！

那么万一我只想要分割 1 个 Primary 与 1 个 Extended 呢？这个时候你的磁盘分割会变成如下所示：



注意到了吗？因为 1~4 号已经被预留下来了，所以第一个 Logical 的代号由 5 号开始计算起来，而后面在被规划的，就以累加的方式增加磁盘代号啰！而其中 /dev/hda3, /dev/hda4 则是空的，被保留下来的代号。

- Linux 底下的硬盘分割模式选择注意事项：
  实际上，在 Linux 安装的时候，已经提供了相当多的预设模式让您选择分割的方式了，不过，无论如何，分割的行为都不是很能符合自己主机的样子！因为毕竟每个人的『想法』都不太一样！因此，强烈建议使用『自订安装，Custom 』这个安装模式！在某些 Linux distribution 中，会将这个模式写的很厉害，叫做是『 Expert, 专家模式 』，这个就厉害了，请相信您自己，了解上面就自称为 专家了吧！没有问题！

  - 自订安装『Custom』：

    - A：初次接触 Linux ：只要切割『 / 』及『 Swap 』即可！
      好了，通常初次安装 Linux 系统的朋友们，我们都会建议他直接以一个最大的扇区『 / 』来安装，这样有个好处，就是不怕分割错误造成无法安装的困境！例如 /usr/ 是 Linux 的可执行程序及相关的文件摆放的目录，所以他的容量需求蛮大的，万一你分割了一块扇区给

/usr ，但是却给的不够大，那么就伤脑筋了！因为会造成无法将数据完全写入的问题，就有可能会无法安装啦！因此上，如果你是初次安装的话，那么可以仅分割成两个扇区『 / 与 Swap 』即可！

- B：建议分割的方法：预留一个备份的扇区！
  就如同前面几个心得分享文章中提到的，由于 Linux 预设的目录是固定的，所以，通常我们会将 /var 及 /home 这两个目录稍微加大一些，如果硬盘够大的话，加个几 GB 也不为过！另外，/usr 至少给他 3~5 GB 吧，如果硬盘真的大的话！而 / 也可以给个几 GB 的空间。最后，由于我们的 Linux 可能是在『试用』阶段，所以很有可能会重复的一再安装，因此上，VBird 都会预留一个扇区来备份我的核心啦与实作过程中觉得不错的 scripts（ 就有点像 DOS 的批次檔 ），当然，我的 /home 底下的咚咚也可以有备份的地方，而安装套件的源文件也可以摆在这里！有个最大的好处是，当我的 Linux 重新安装的时候，我的一些套件马上就可以直接在硬盘当中找到！呵呵！重新安装比较便利啦！

- 选择 Linux 安装程序提供的的硬盘分割方式：
  对于首次接触 Linux 的朋友们，通常不建议使用各个 distribution 所提供预设的 Server 安装方式，因为会让你无法得知 Linux 在搞什么鬼，而且也不见得可以符合你的需求！注意：选择 Server 的时候，请『确定』您的硬盘数据是不要的！因为 Linux 会自动的把你的硬盘里面旧有的数据全部杀掉！此外，硬盘至少需要 2 GB 以上才可以选择这一个模式！

硬盘方面的规划大致上就是如此啦！要规划硬盘的时候，请特别的小心呦！

Linux 安装前准备

Linux 安装之前要准备什么呢？就是刚刚前面已经讲过的几个咚咚啦！归纳一下：

1. Linux 主机规划单：就是刚刚我们规划好的那个单据啰！
2. Linux distribution ：利用一些映象站台下载各版本的 Linux ，或者直接以本书提供的三块 CD 装的 Mandrake 进行安装啰！
3. 主机硬件信息收集：根据主机规划单的内容，去收集一下你的硬件信息吧！其中特别重要的是，先检查一下是否可以使用光盘开机呢？如果 BIOS 不能支持光盘开机的话，那么就需要先行安装可开机软盘。
4. 网络硬件联机：这部份本书先不谈，否则内容就太多了，阿！再写下去鸟哥会疯掉……所以请大家先上网查阅一下网络的硬件联机吧！
5. 网络信息：包括你的 IP, netmask, gateway, dns IP 、是否为拨接等等，都需要先知道呦！

然后，其实各个套件的安装步骤都差不多，大概都是：

- A. 选择安装模式：主要分为图形接口安装与文字接口安装；如果是图形接口安装的话，还可以选择语系，这个时候我们就有中文可以使用啦！
- B. 搜寻硬件信息：然后安装程序会去搜寻一下系统的硬设备，以利后续的处理，有的安装程序会在这个地方让您加入一些参数，以驱动不明的装置设备；
- C. 设定键盘、鼠标模式：这个可是很重要的项目呀！
- D. 硬盘分割设定：就是刚刚提到的几个注意事项；
- E. 套件选择：这是很重要的部分呢！请特别留意！
- F. 网络与安全性设定：连上 Internet 的模式与驱动网络卡的方式等设定；
- G. 超级管理员与一般身份使用者账号设定：最重要的是设定 root (超级管理员) 的密码啰！
- H. 设定 X-Window 与开机片：如果有安装 X-Window 相关的软件，那么 X-Window 就需要设定并且测试一下，另外，制作开机片永远是最正确的选择！

大概就是这样子吧！好了，底下我们就真的要来安装啰！

一个 Linux 安装实例
好咯，我要开始安装我的 Linux 啦！那么先说明一下我的基本规划：

- Linux 主机定位：
  关于这部 Linux 主机，主要是用来做为练习与比较各不同版本 Linux 之间的差异用的，所以定位在练习上面，预计并不提供任何的网络服务；

- 硬件要求：
  因为定位在练习上面，所以以手边有的机器来做为练习就是了，且由于是手边的工作机，所以必须要安装多重开机系统。

  - CPU 使用 P-III 933 的 CPU ，这个是我的工作机啦，实际上不需要这么高档的货色；
  - 内存使用 128 MB ，是 PC 133 的规格；
  - 硬盘使用手边有的一颗 30 GB 硬盘，预计未来要在上面安装 3 ~ 4 个 Linux 系统，所以在这次的安装中，预计规划 /, /home, swap 三个磁盘区块而已；
  - 网络卡使用最常见的螃蟹卡；
  - 显示卡则是前一阵子的主流，但是目前已经落伍了的 Geforce 2 MX ；
  - 其它软盘机、光驱、鼠标、键盘等等的配备，则是一般的个人计算机之配备

- 磁盘分割：
  如同刚刚提到的，由于是定位在练习上面，所以仅分割出 /, /home, swap 三个磁盘区块，各别占约：

  - /　　　/dev/hda1 ：4 GB
  - /home　/dev/hda3 ：1 GB
  - Swap　 /dev/hda2 ：200 MB
  - 其它则为尚未规划空间

- 由于需要多重开机，所以选择开机管理程序为 grub 这个程序，并且安装在 MBR 里头；
- 套件选择：由于是定位在练习上面，并且不提供任何的网络服务，所以一些零碎的套件将不安装，并且 X-Window 仅安装个人较为喜好的 KDE 套件；

好啦！真的要来安装了！请依照下面的步骤来进行吧！

1. 选择开机次序并开机：
   我们可以在 BIOS 里面设定开机的次序，看看能不能以 光驱 开机！现今的主机板大多已经支持这项功能了，而我们的 Mandrake 9.0 光盘片本身就是可以开机的，所以设定成光盘开机最好啰！设定方式为：

   - 按电源键开机；
   - 在进入系统之前会出现 Del 字样（每个厂牌不太相同），此时按下键盘上的 Delete 键；
   - 进入 BIOS 之后以方向键选择 『BIOS Features Setup』这一项，或者是 『Advanced BIOS Features』，不管如何，反正只要看到 『BIOS Features』字样的那一项就对了！；
   - 将方向键移动至 『Boot Sequence』 或者是 『First Boot Device』； 这一项，按键盘上的『Page Up』 或 『Page Down』 按键，选择『CD-ROM』 为第一开机顺位即可。这里注意一下，如果你的机器并不支持 CD-ROM 开机的话，你一定找不到 CD-ROM 这一项，这时请制作开机片吧，并将此项调整为『A』为第一顺位；
   - 按键盘上 『ESC』 键退出；
   - 将方向键移动至 『Save and Exit』 这一项按 『Enter』 及 『Y』 确认后重新开机即可！

   如果是必须以软盘开机的话，那么需要的动作就变成了：

   - 随便找一台 Windows 计算机，启动档案总管，进入光驱的档案数据夹，假如您的光驱在 E 槽，那么请进入 E:\dosutils 这个目录，请注意，每个人的光驱所在磁盘代号都不一样，请依您的计算机来操作；
   - 在该目录当中，点选 rawrite 这个应用程序，然后在出现的 MS-DOS 画面之中依序输入下列：（ 注：那个磁盘代号是光驱，请依您的计算机实际配置来决定！ ）

   ```
   Enter disk image source file name: e:\images\cdrom.img
   Enter target diskette drive: a:
   Please insert a formatted diskette into drive A: and press -ENTER- :
   ```

   -

   请注意，请将软盘放入您的软盘机当中呦！

   无论如何，在进行完上面的步骤之后，请将第一片 Mandrake 可开机光盘放入光驱中，如果使用

软盘的朋友，请将可开机软盘放入软盘机，否则拿出软盘，按下电源，给他开机去！

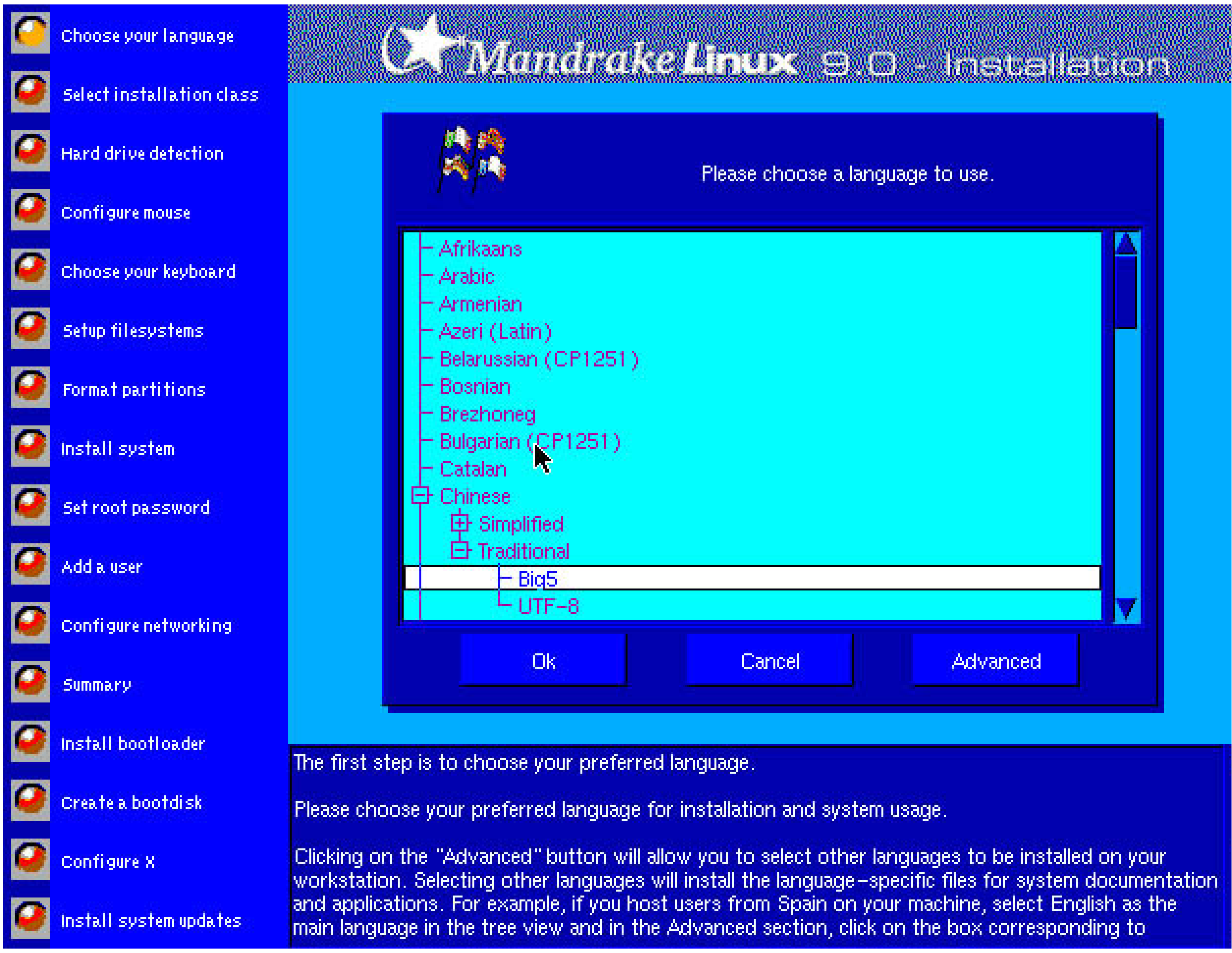
2. 选择安装模式：

一般 Linux 支持两种安装模式，分别是图形接口与文字接口。正常的话，在开机之后，会到底下的这个画面，在这里可以直接按下 Enter 来进入图形安装接口，或者在 boot: 后面输入 text 来以文字接口安装。由于 Mandrake 的安装程序做的蛮好的，所以通常可以顺利的进入图形安装程序当中。如果必须以文字接口来安装的话，那么也没有关系，因为全部的步骤都跟图形接口下一模一样，所以您可以对照着这个网页的步骤来试看看。



3. 选择安装程序的语系：

在 boot: 之后，会跑一些安装程序所需要的数据，然后就是进入这个语系选择的画面。由于我们比较看的懂中文呀！所以，在进入图形接口之后，请移动鼠标并且选择上面的项目，选择完毕之后，请按下 OK 按钮即可；



4. 是否接受授权码规定：

授权码一定要选择接受才可以继续呦！所以就接受吧！同时请注意到，在这个安装程序的画面中，主要分为三个区域：

- 执行流程步骤区：这个区域是在左边的流程列，您会发现到上面的画面中，那个『选择语系』左边的按钮列颜色不一样！对啦，那表示『已经或正在安装的步骤』咯！而在下方的颜色则是表示尚待进行中的流程。好了，那么假设您已经进行到了第五个流程，亦即是『选择键盘形式』那个流程时，却想要回到前一个流程，亦即是『设定鼠标』时，可以将鼠标移动到『设定鼠标』左边的按钮，按下他，嘿！就回到设定鼠标的画面啦！

- 此步骤的提示内容：在上头画面的右下方，就是此一步骤的提示协助文字区(Help)，您可以到该画面的右边滚动条处移动，以了解完整的信息；

- 该步骤的选择项目：就是占了画面最大面积的那个框框当中啦！里面是关于该步骤安装时，需要您来设定的选择项目，请仔细的进行选择吧！

此外，在某些流程步骤当中，会另外有跳出式窗口来提供您选择或设定，这就是基本的安装程序画面啰。



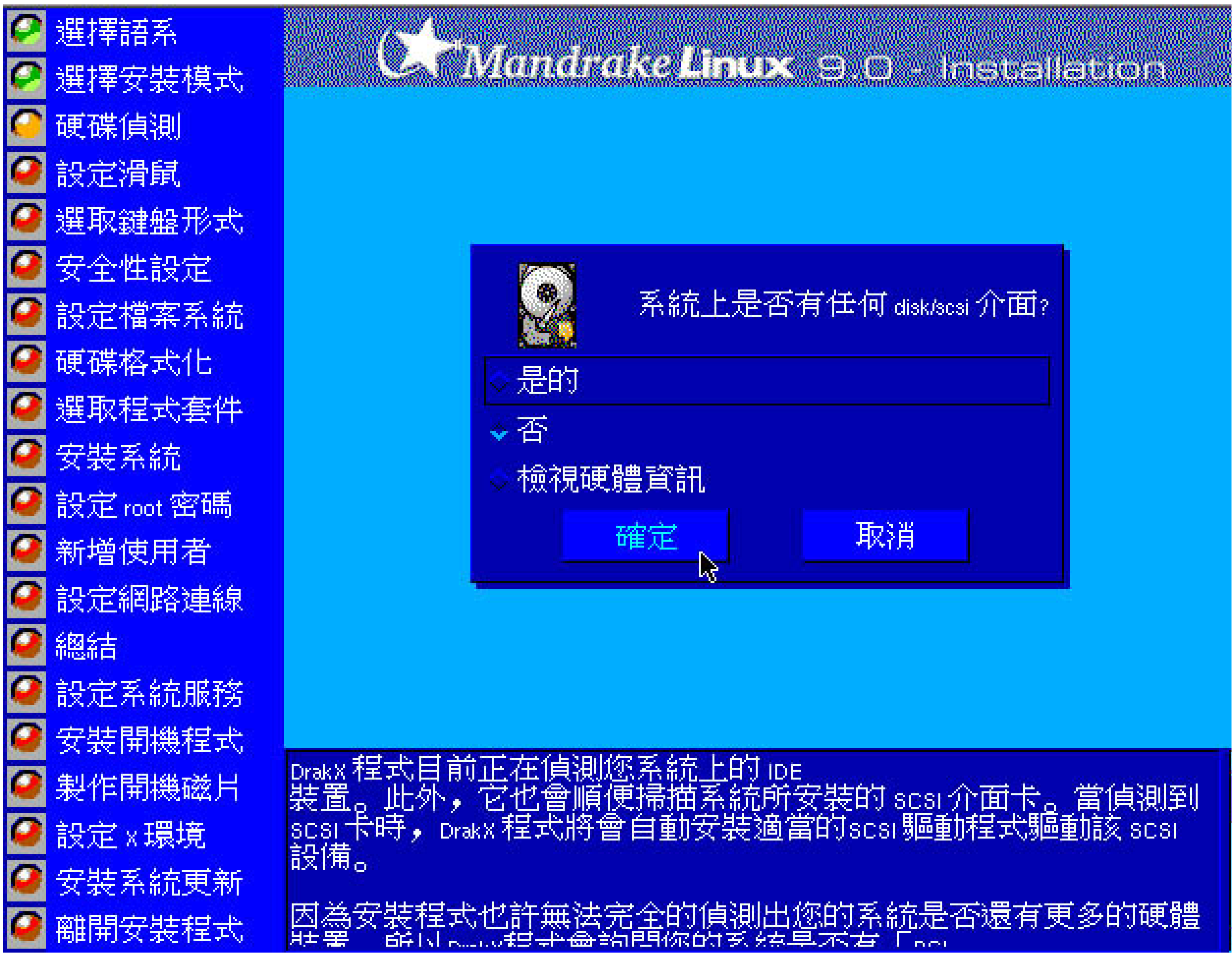
5. 选择预设或自订安装：

目前的 distribution 通常还蛮好心的，会询问您是否要『安装』还是在『既有的 Linux 上面升级』，另外，也可以不升级核心，仅升级可以升级的套件！由于我们是第一次安装，且想要以自己最想要的方式来安装，所以当然就如同上面一般，选择『自订』及按下『安装』即可！



6. 硬盘侦测：

由于您的系统上面可能会有 SCSI 接口的硬盘，果真有的话，由于 Linux 会再以额外的程序去侦测并驱动 SCSI 接口的硬盘，所以这里才需要选择『是的』，如果您跟 VBird 一样是个苦命的人，那么这里直接给他『否』吧！当然，有兴趣的话，可以去『检视硬件信息』看看你的硬件配备是否被正常的驱动了呢？



7. 设定鼠标：

选择属于您正确的鼠标类型，这里设定错的话，有些 distribution 可是不许您进入 X-Window 的呦！鼠标比较有关系的是他的连接到主机的类型啦！如果是圆头的，那就称为是『PS2 接口』的鼠标，如果是有九个孔的，就称为『串行端口』鼠标，如果是扁头的，那就是『USB』鼠标啦！我们这里选择标准就好了！



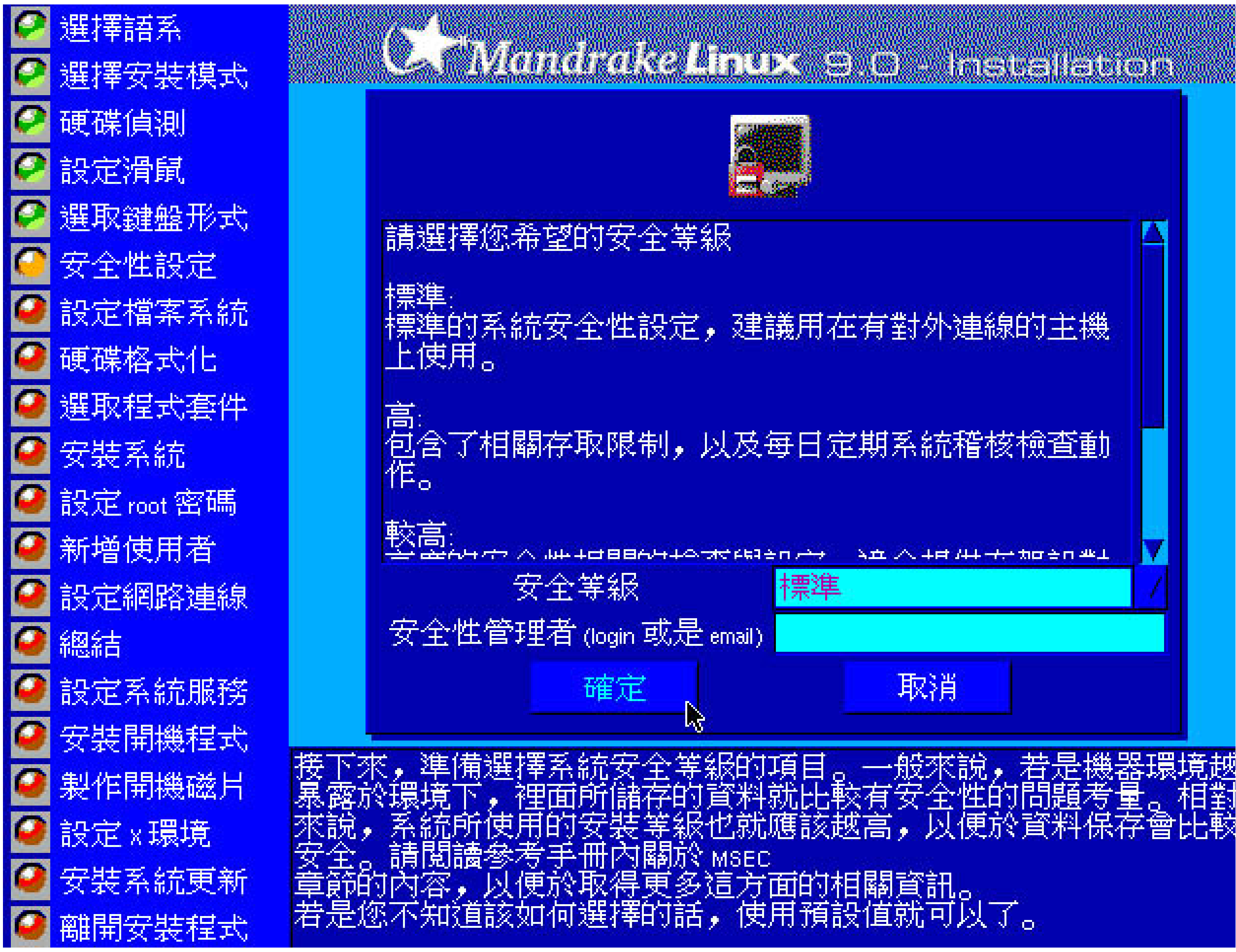
8. 设定键盘：

通常我们直接选择美式键盘即可！



9. 安全性设定：

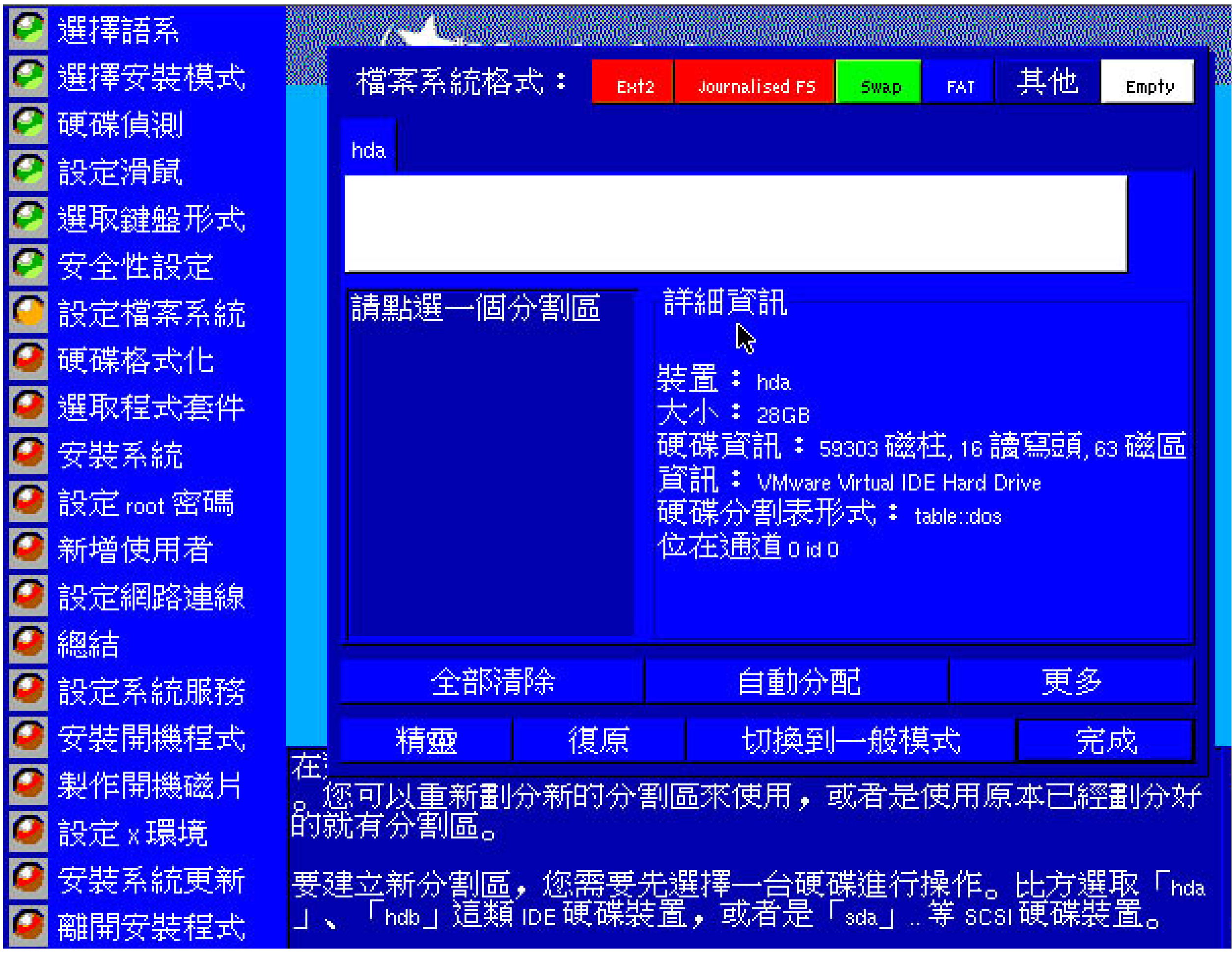
共分为四种等级，分别为『标准』、『高』、『较高』、『严密』等四个，通常我们选择『标准』就可以了！对于安全性而言，这样的设定可能并不足够，需要再进一步的设定，关于安全性的设定文章，请参考相关书籍。不过，我们这里的定义是练习用主机，所以选择『标准』即可。



10. 设定档案系统及硬盘分割：

接着下来是最重要的硬盘分割方式！如上图所示，我们会看到硬盘的整体信息，由于我们的硬盘尚未规划，所以在 hda 的部分为白色的模样！此外，在底下有几个重要的按钮，分别的功能是：

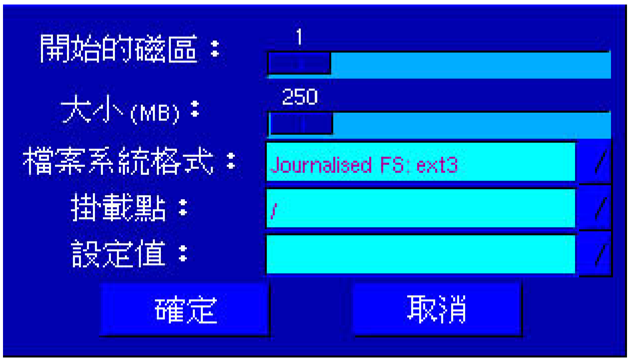
- 全部清除：将原有的硬盘分割全部删除成未分割的样子；
- 自动分配：以系统磁盘分割方式进行磁盘分割；
- 更多：更多的详细资料；
- 精灵：叫出提示精灵；
- 复原：恢复成尚未进行分割前的样子；
- 切换到一般模式：换成较为简易的设定画面；
- 完成：完成磁盘分割，并将 partition table 写入 MBR 内。


選擇語系
選擇安裝模式
硬碟偵測
設定滑鼠
選取鍵盤形式
安全性設定
設定檔案系統
硬碟格式化
選取程式套件
安裝系統
設定 root 密碼
新增使用者
設定網路連線
總結
設定系統服務
安裝開機程式
製作開機磁片
設定 x 環境
安裝系統更新
離開安裝程式
檔案系統格式：
Ext2
Journalised FS
Swap
FAT
其他
Empty
hda
請點選一個分割區
詳細資訊
裝置：hda
大小：28GB
硬碟資訊：59303 磁柱, 16 讀寫頭, 63 磁區
資訊：VMware Virtual IDE Hard Drive
硬碟分割表形式：table::dos
位在通道 0 id 0
全部清除
自動分配
更多
精靈
復原
切換到一般模式
完成
。您可以重新劃分新的分割區來使用，或者是使用原本已經劃分好的就有分割區。
要建立新分割區，您需要先選擇一台硬碟進行操作。比方選取「hda」、「hdb」這類 IDE 硬碟裝置，或者是「sda」.. 等 SCSI 硬碟裝置。

- A. 建立根目录分割：



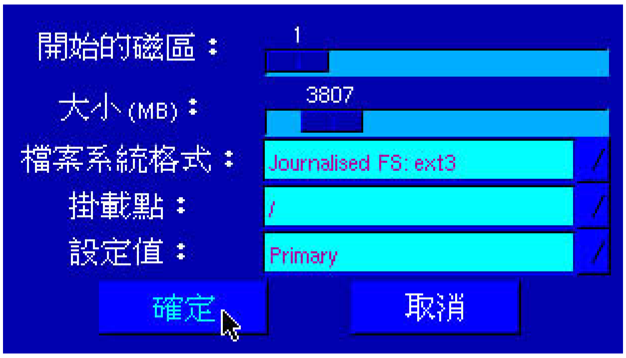
在按一下 hda 那个空白的区域之后，在选择动作栏内会出现『新建』字样，按下『新建』会出现底下的图样：



在这个跳出式的窗口之中，我们要选择的是开始的扇区与大小，刚刚我们的设定是需要根目录约 4 GB 左右，这里『开始的扇区』使用默认值即可，鼠标移到『大小』右边的那个滚动条上面，请注意，由于每个磁盘的大小不同，所以一个磁柱的大小也都不一样，因此，你的屏幕前面看到的内容可能与我的不一样，请特别留意！



设定完了磁盘的大小之后，再来是要选择磁盘的档案格式，Linux 预设的档案格式是 Ext2 ，但是更新的 Ext3 档案格式中，提供了更多的日志式记录功能，所以目前也可以使用 Ext3 这一个档案格式哟!同时,我们选择了『 / 』根目录做为挂载点以及『Primary』做为设定值，所以，最后得到的结果为：



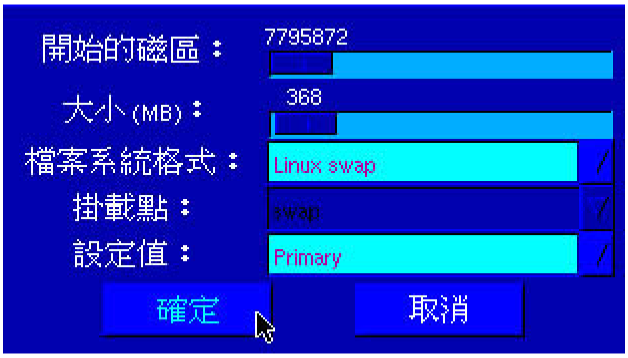
再按下『确定』之后，就会显示出目前这个扇区的属性啦！如下图所示：



在上面的图标中，将鼠标移动到空白的区域之内，按一下鼠标左键，就又会出现『新建』的字样，此时，再继续新建的工作！好啦，我们来新建一下 Swap 这个虚拟内存吧！

- B. 建立虚拟内存 Swap ：

  与新建的功能相当，不过，需要选择的则是『档案系统格式』内容，需要选择为『 Linux swap 』的格式，则底下『挂载点』会自动被取消掉！

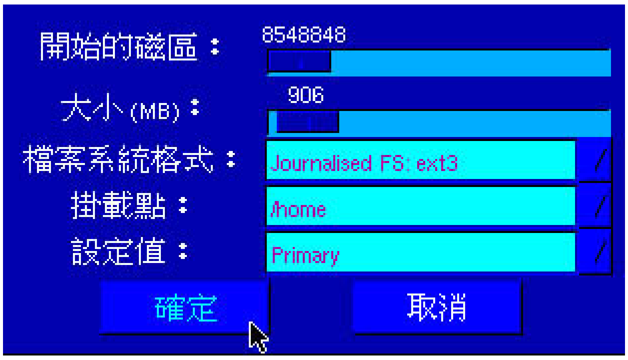


这个 Swap 有什么功能呢？简单的说，他可以被看做为『虚拟内存』啰，那么虚拟内存是什么？您可以这样想象，当你的物理内存只有 64 MB 的时候，但是你的系统负荷突然之间太大了，例如突然之间有数十个人连上你的 Web 服务器时，那么你的物理内存将不足以负荷这些计算的数据！怎么办？这个时候我们可以使用硬盘来仿真内存的数据存取，这个就是所谓的『虚拟内存』啰！不过，虚拟内存的速度会比较慢哟！

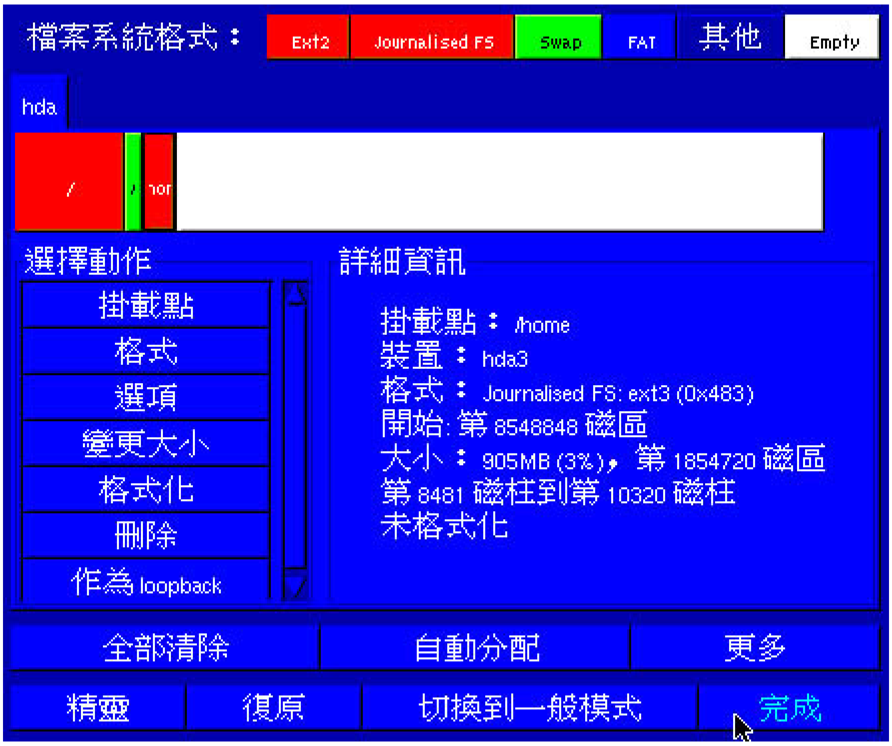
当有数据被存放在物理内存里面，但是这些数据又不是常被 CPU 所取用时，那么这些不常被使用的程序将会被丢到虚拟内存当中，而将速度较快的物理内存空间释放出来给真正需要的程序使用！这就是虚拟内存 的功效啦！通常 Swap 建议的值大约是『RAM 的两倍大』，但是这个因地制宜啦！像我的 Proxy 主机本身的内存就达到 1GB 了，那么是

否还需要虚拟内存呢？见仁见智啰！

- C. 新增其它挂载扇区：



好啦！再来以同样的方法建立其它的磁盘分割，同样的方式建立起 /home 这一个磁盘分割吧！请注意，上面三个我都使用 Primary 进行分割的哟！最后的数据就成为：



- D. 使用预设分割行为分割：



安装程序也提供了三种主要的预设分割方式来给使用者，当我按下『全部清除』并且再按下『自动分配』之后，会出现上面的窗口，其中，他们的分割方式分别为：

- with /usr：

  / ：1GB
  Swap ：2 * RAM
  /usr ：3.9 GB
  /home ：其它剩余的空间都直接给 /home

- simple：

  / ：5.3 GB
  Swap ：2 * RAM
  /home ：其它剩余的空间

- server：

  / ：256 MB
  Swap ：3 * RAM
  /usr ：3.9 GB
  /tmp ：500 MB
  /var 与 /home 平均分配其它的硬盘空间。

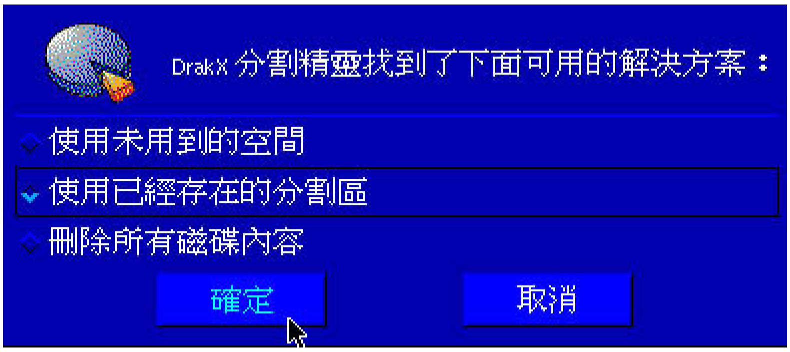
- 同样的，我们这里不建议使用安装程序提供的方式来分割啦！

- E. 写入磁盘分割表：

  最后，就给他输入『完成』，并且在出现的窗口中，将硬盘分割表写入，这样就完成了我们的硬盘分割啰！哇！好累呦！

- F. 选择刚刚分割完成的磁盘分割表：

  使用刚刚完成的那个分割表，所以选择第二项即可按下确定！

- G. 设定挂载点：

  再次的设定挂载点，对应好刚刚的分割表！嘿嘿！这样就完成了最麻烦的工作之一了！

- H. 硬盘分割的建议：

  这里必须要给 Linux 新鲜人一些硬盘分割上面的建议：

  - 甲、 初次使用 Linux ：

    Swap 约 100 MB
    其它的都给 / ；

  - 乙、 进阶使用者：

    Swap 约 100 MB;
    /var 给 3?5 GB;
    /usr 给 3?5 GB;
    / 给 1 GB 以上;

/home 可以给大一些；
/backup 用来做为备份的扇区

11. 硬盘格式化：

接着下来，硬盘分割完毕之后，就是格式化硬盘啦！没错，所以这里选择一下你要格式化的磁盘，当然，如果该磁盘早就存在，那么不格式化也没有关系！ ^_^….

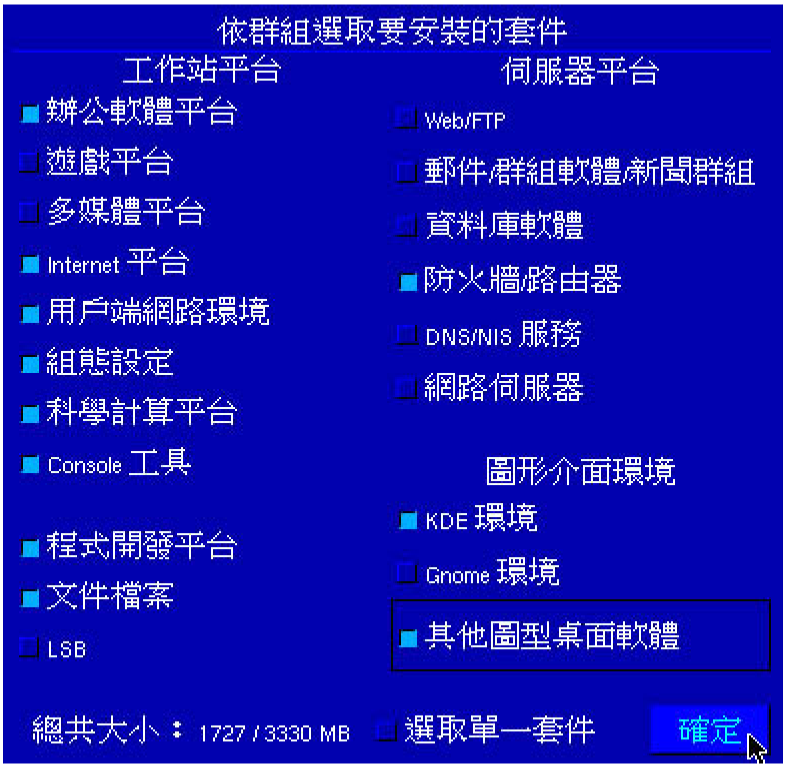


12. 选取程序套件：

接着下来自然就是选择套件啰！而要选择套件之前， Mandrake 会先检查是否有这些安装套件的原始码存在，所以会先显示是否有上面这些光盘片存在？如果您使用的是书上附的光盘片，那么就会出现上面的内容咯！请勾选他吧！


選擇語系
選擇安裝模式
硬碟偵測
設定滑鼠
選取鍵盤形式
設定網路連線
總結
設定系統服務
安裝開機程式
製作開機磁片
設定 x 環境
安裝系統更新
離開安裝程式
Mandrake Linux 9.0 - Installation
如果您有下面清單內所列出的 CD 光碟片，請選擇確定。
如果都沒有的話，請選擇取消。
標為 "Installation CD 2 (x86)" 的光碟片
標為 "International CD (x86)" 的光碟片
確定
取消

o A. 选择所需要的套件：



接下来我们要来选择的就是所需要安装的套件啦！刚刚已经提过了，我们需要的是 KDE 与相关的 X-Window 套件，另外，也需要几乎所有的练习用的资料，所以可以选择如上面所示的套件项目，可以在上图的左下角发现选择的套件容量为 1727 MB ！

上表中，最需要注意的是『Console 工具』、『程序开发平台』及『文件档案』这几个东西，如果我们未来在进行一些程序编译的时候，或者是自己在网络上面下载软件来安装的时候，常常会使用到这三个项目内的数据，如果没有选择的话，会死翘翘！

底下提供五个建议来给大家做为套件选择上面的考虑：

- 预设给桌上型计算机用的：这是默认值共需 969 MB 的空间，缺点是没有 compiler ，以后不能以原始码安装软件。

  办公室软件平台
  Internet 平台
  KDE 环境
  Gnome 环境

o

- 全部都安装：全部安装需要大约 2193 MB 的空间，特别建议新手以这个方式来安装您的第一套 Linux ，以降低学习的困难呢！

- 小硬盘的选择：没有 X-Window, 并且未来也不能自行安装软件，不过就是可以节省硬盘空间，占了大约 430 MB 的容量。

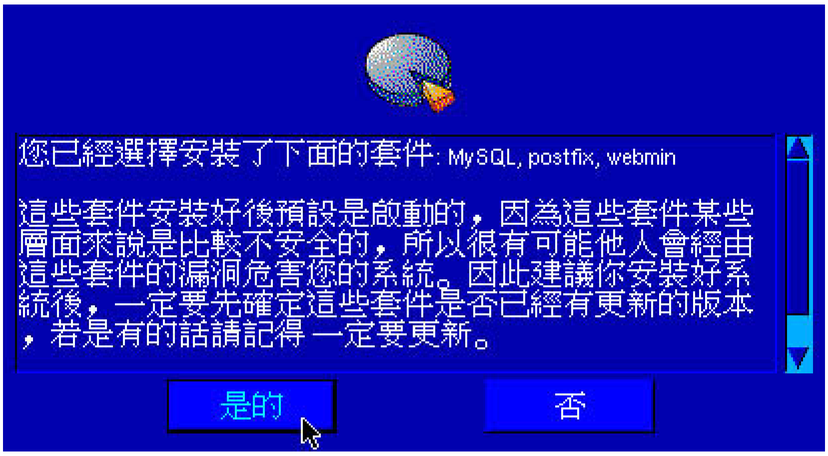
  客户端网络环境
  组态设定
  Console 工具
  文件档案

- 只有文字接口的服务器用主机：特别适合用来进行架站的选择套件方式，没有 X-Window 呦，而且仅占容量为 958 MB，是鸟哥最喜欢的安装方式了！

  Internet 平台
  客户端网络环境
  组态设定
  Console 工具
  程序开发平台(特别重要，一定要选择)
  文件档案
  防火墙、路由器

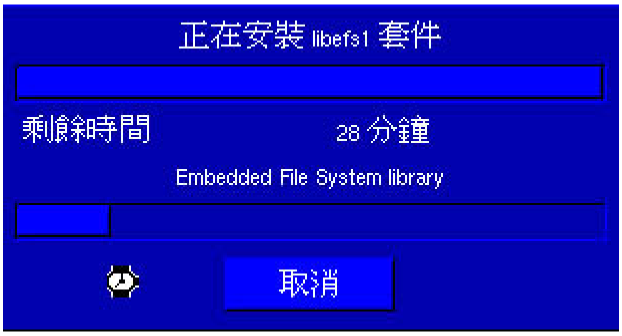
- 加上 X-Window 的服务器用主机：这个比上一个多了 X-window 就是了，共享了 1727 MB，也是我们上面的图示的内容！建议第二次玩 Linux 的朋友，并且还是想要使用 X-Window 的朋友安装！

o B. 危险套件的警告标语：



当你选择了一些套件，但是这些套件在 Linux 上面可能不是这么安全，所以 Mandrake 特别提出告示警语，呵呵！没关系，我们早就知道会有这些咚咚，所以不用理他吧！

o C. 开始正式安装啰：



努力的给他安装中……喝杯茶，看个电视去？不过，在安装的过程中，会要求您换片，依序换上第二片 CD ，以及第三片名为 Internal CD 的，安装完毕之后，就可以进入到下一个画面了！

13. 设定 root 密码：

这个也是相当重要的呦！那就是系统管理员的密码啦！由于我们预设使用 Linux 本身的密码机制，所以选择 本机密码 数据，至于其它的密码格式，则请参考个别的网络书籍吧！

对于密码的设定，如果您的机器可能会上网，那么不论是否为 Server ，最好将 root 的密码设定的严格一点，例如至少 8 个字符以上，而且含有特殊符号，例如： I&my_dog 之类的怪怪密码！不但不容易被猜测，自己还蛮容易记忆的为主！



14. 预设使用者登入系统：

如果您的系统预计是要给许多人来使用的，那么这个『开机预设使用者身份登入』的功能最好是取消，对您而言会比较有保障！



15. 新增使用者：

您可以在这个动作里面设定你的一般身份使用者的账号与密码，也可以在未来进行账号与密码的设定，这里我们先设定一个名为 test 的使用者，并且帮他设定一下密码，设定完成之后，先按下『接受使用者』，再按下『完成』，就可以进入到下一步骤啰！至于其它的使用者设定，我们将在后面的章节再介绍。

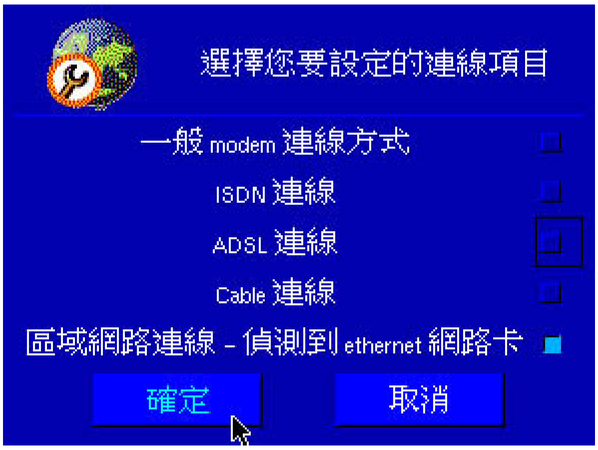


16. 设定网络：

很多朋友对于设定网络的问题都很困扰耶！这真是伤脑筋了！不过没有关系，我们底下以一般性的用法来设定你的网络卡，如果你不知道如何设定你的网络卡，那么就直接以底下的例子来设定吧！



- A. 选择自动侦测网络卡与联机模式：



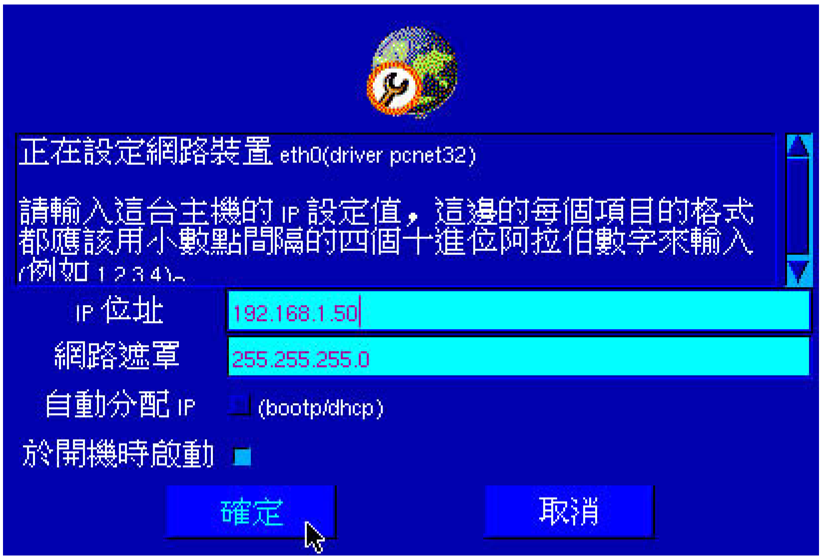
选择自动侦测网络卡之后，如果能够见到上面的图示您应该要觉得『哇！好高兴！』因为看到这个画面表示你的网络卡已经被 Linux 捉到了！这还不够高兴呀！应该要很爽的哩！好了，接着下来设定其它的东西吧！按下确定；

- B. 网络适配卡信息选择：



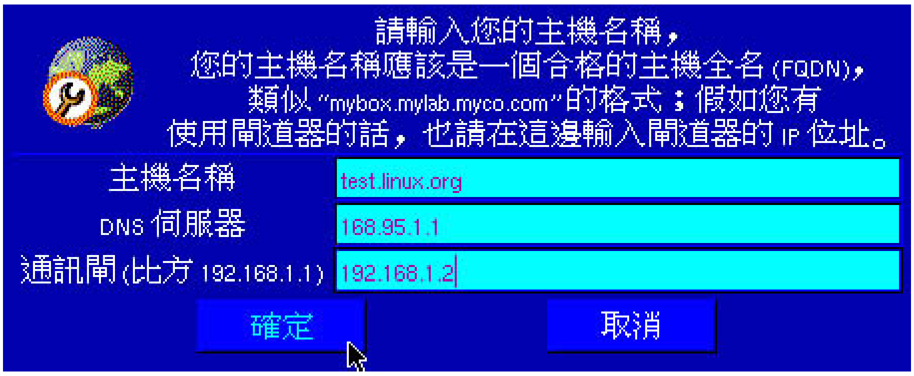
注意一下上面显示的是否为你的网络配备，没有问题的话，那么就给他按下『否』之后，选择确定吧！

- C. 设定网络卡地址 IP：



网络卡地址(IP)选择最简单的私有 IP 来设定即可！如果不知道如何设定，那么就设定上面的样子就好了！

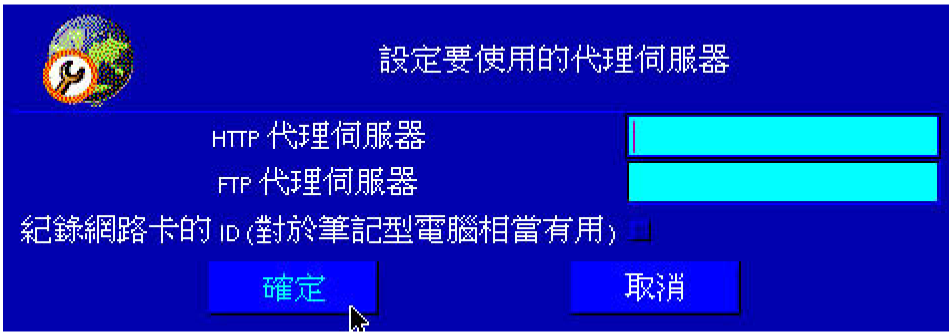
- D. 设定主机名称与 DNS 相关：



在实际的网络世界中，每一部主机都有他『独一无二的名字』，那就是这里设定的主机名称啦！不过，因为我们没有对外公布我们的主机名称，所以这里随便你设定没有关系啦！相关的主机名称讯息，请参考 DNS (Domain name server) 的相关文章。至于 DNS

服务器与通讯闸，那个 DNS 服务器可以直接填中华电信的，168.95.1.1 大家都能使用，没有问题，再来的那个通讯闸就不见得每个人都一样了！如果你还是不知道你的网络状况，那么还是先跟我一样的填法吧未来可以自行修改呢！

- E. 使用代理服务器：



如果不知道你的代理服务器是什么，就不要设定吧！没有关系的啦！

17. 设定总结：
好啦！看看有没有设定错误啦！没有的话就给他『确定』下去吧！



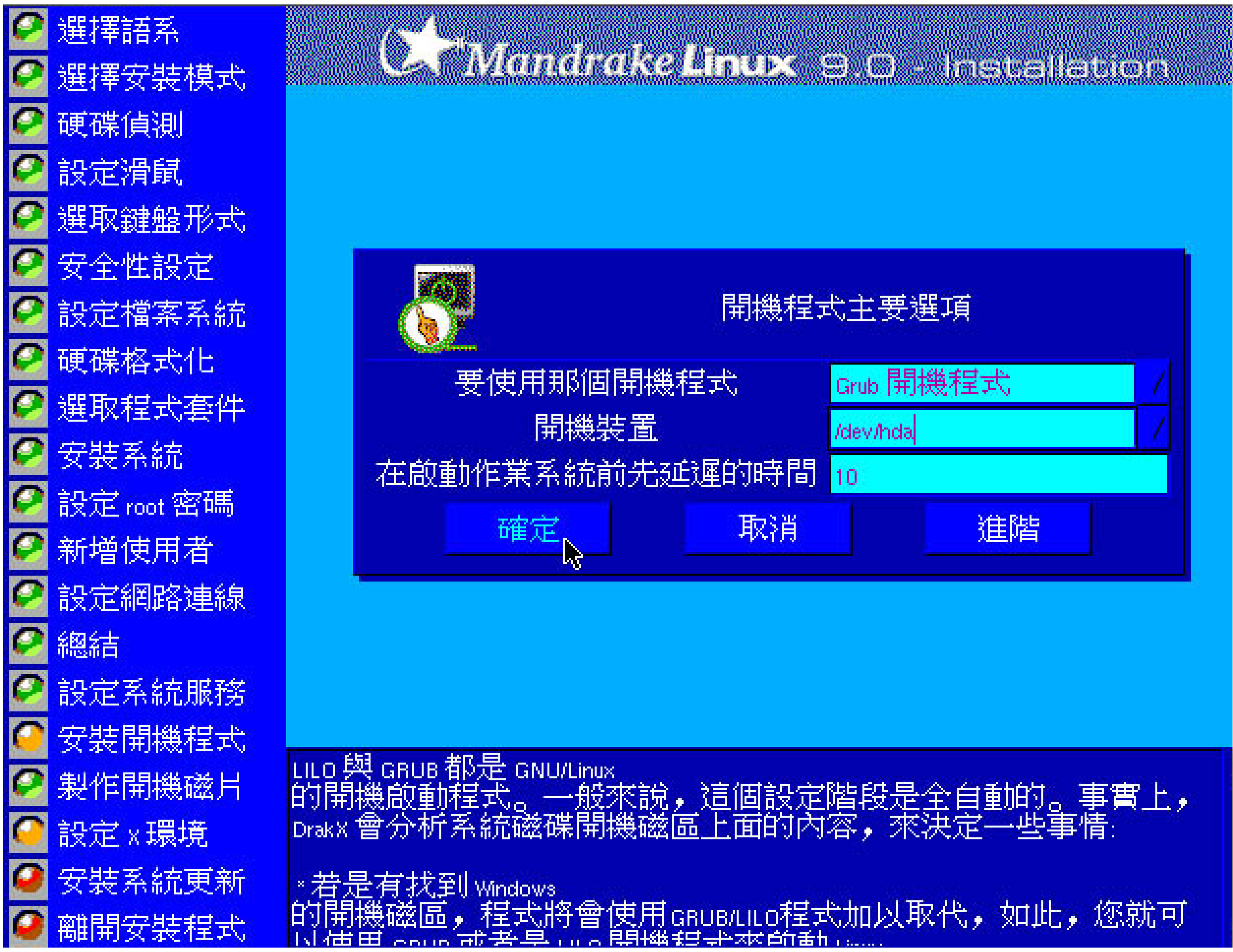
18. 设定系统服务：

在预设的状态之下，系统就会启动一些服务来自我调整使用环境，这些包含了登录档案的纪录、例行命令的执行与内存管理等等，这个部分我们会在后面的章节继续说明，所以也可以直接给他『确定』下去吧！



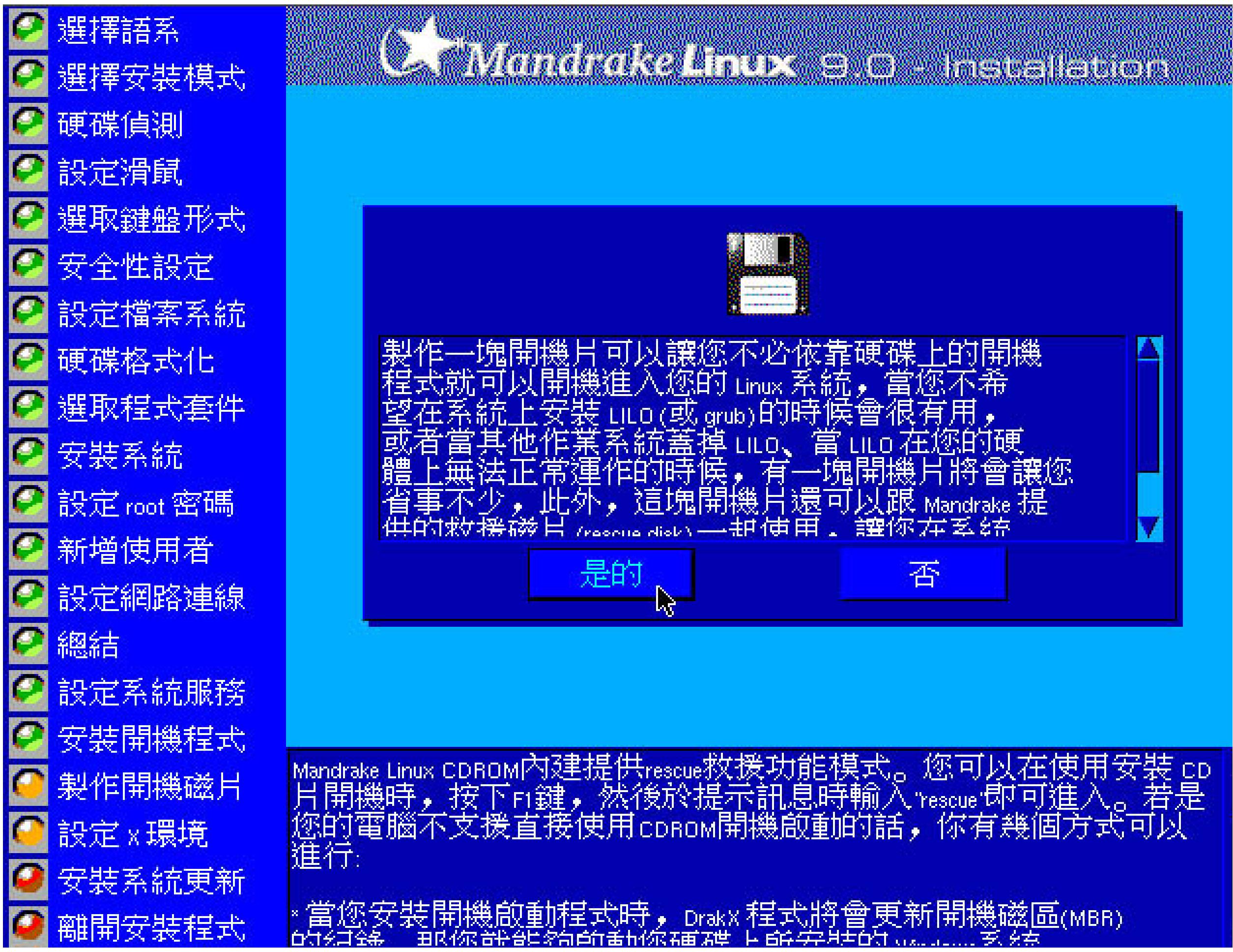
19. 设定开机管理程序：

我们以较新的 Grub 开机管理程序来管理我们的 Linux 主机吧！同时，将他安装在开机扇区的 MBR 里面，作业延迟表示『选单会停留 10 秒钟』来让我们决定是否要以该核心进入 Linux 系统！



20. 建立软盘开机片：

无论任何时刻，建立可以开机的救援磁盘都是一件正确的选择！因为您无法肯定什么时候会来个全台大停电，您总不希望努力了这么久的安装好了的系统被破坏吧！ ^_^””



21. 设定 X-Window ：

如果您有安装关于 X-Window 这个窗口接口的软件，例如 KDE, Gnome 等等的咚咚时，那么就会出现这个设定 X 环境的选项啰！『设定窗口接口是个很麻烦的程序，而且一定会不成功』，这是因为 Mandrake 的安装程序在设定 X-Window System 的部分有点小问题！所以，『请不要测试 X-Window 』！然而，无论如何，如果设定不成功，仍然可以在装完成之后再进行重新设定，所以这一步即使设定错误了，也别难过，我们在『系统管理员篇』的时候，会再详细的介绍 X-Window 的设定方法！

同时也请特别留意， X-Window 在 Linux 里面『仅是一套软件』，而且他还是相当有趣的一套软件，怎么说呢？ X-Window 又分为两部份，第一部份称为 X Server ，这个 X Server 负责 Linux 主机硬件的管理，例如显示卡、鼠标、键盘、屏幕分辨率等等，都是他在管，而这个 X Server 即是鼎鼎大名的『 XFree86 』是也，而负责整个桌面的显示的管理者，就称为 Window Manager (窗口管理员) 软件啰，目前最让大家熟知的就是 KDE 与 GNOME 这两套窗口管理系统啦！那么也就是说，只要您的 XFree86 死掉的话，那么 KDE 也好， GNOME 也好，就肯定也是死掉的，而我们通常说的在『设定 X-Window 』通常就是设定那个 XFree86 啰，包括屏幕的分辨率、更新频率等等的！



- A. 设定分辨率：

  分辨率的设定就如同上面所示，一般使用者计算机的分辨率设定大都是『 800x600 更新频率 60-80 』，但是鸟哥特别喜欢字小小的，桌面大大的，所以我会选择 1024 x 768 那个分辨率呢！

- B. 选择显示卡芯片组：



刚刚提过，我的显示卡为 Geforce 2 mx ，所以选择这一个就对啦！『看！』之前不是跟您说过最好先了解一下自己的硬件吗？嘿嘿！就是这些地方用的到啦！

- C. 选择 X Server 版本：



目前 XFree86 分为两个版本，较新的是 4.x.x 版，旧版的为 3.3.x 版，由于 4.x.x 支持度比较高，当然选择 4.2.1 那个版本啰！

- D. 调整色彩度：



调整一下您的未来的桌面吧！这里我选择我最喜欢的大小，您亦可选择您喜欢的色彩呦！

- E. 测试设定值：

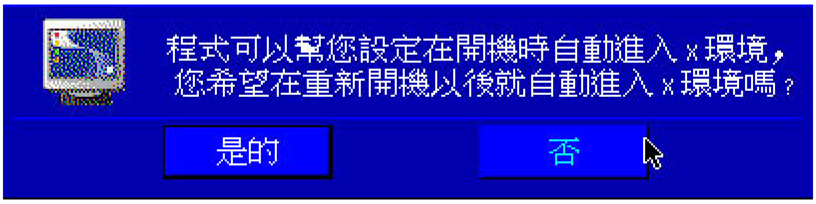


记住喔！由于 Mandrake 的安装程序问题，这个测试的功能『一定会失败！』，所以请不要测试喔！但是，如果不小心测试下去了，屏幕变成黑压压的一片时，怎么办？！别担心，这个时候给他按下：

[Ctrl] + [Alt] + [F1]

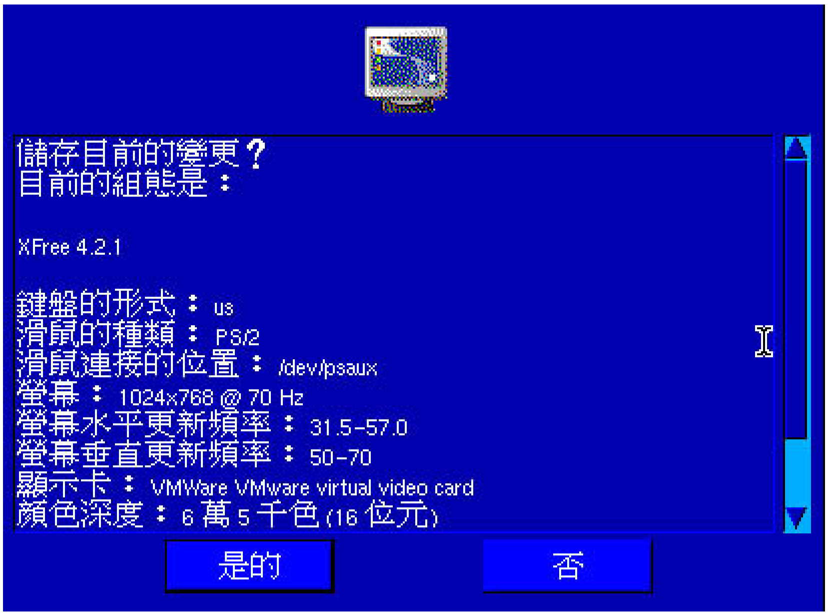
就会回到刚刚的画面了！无论如何，如果测试成功的话，那么就会进入到下一个画面。

- F. 选择是否以 X-Window 做为预设的开机登入系统：



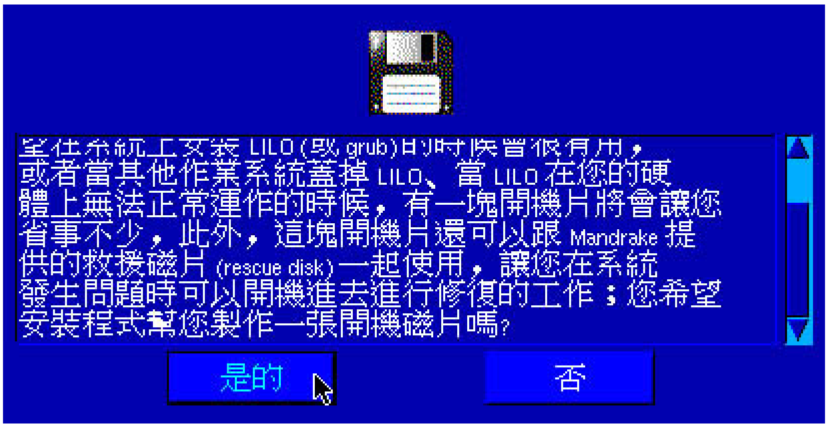
就像之前就一直在说的，不需要直接进入 X-Window 啦！反正在文字接口之下，仍然可以轻松的就进入 X-Window 的说！所以这里我是选择『否』呦！

- G. 再次确认 X Server 设定值：



如果没有问题的话，就给他『是的』下去吧！

- H. 制作救援磁盘：



我也不是很清楚为什么这里还要再制作一次救援磁盘？无论如何，小心驶得万年船，所以还是再拿另外一块磁盘来制作一次吧！这样 X-Window 就设定完成啰！

22. 套件修补中心：

这可不是坐月子中心呐！因为发展出来的各个 Linux 套件都很有可能被一些网络闲人所破解，而利用一些漏洞来影响或破坏别人家的主机，因此，必须要常常进行各种套件修补的动作！不过，由于我们的网络还没有搞定，所以….这个步骤也就没有办法做啦！选择『否』吧！等到安装完毕，并且连上 Internet 之后，再来修补漏洞！



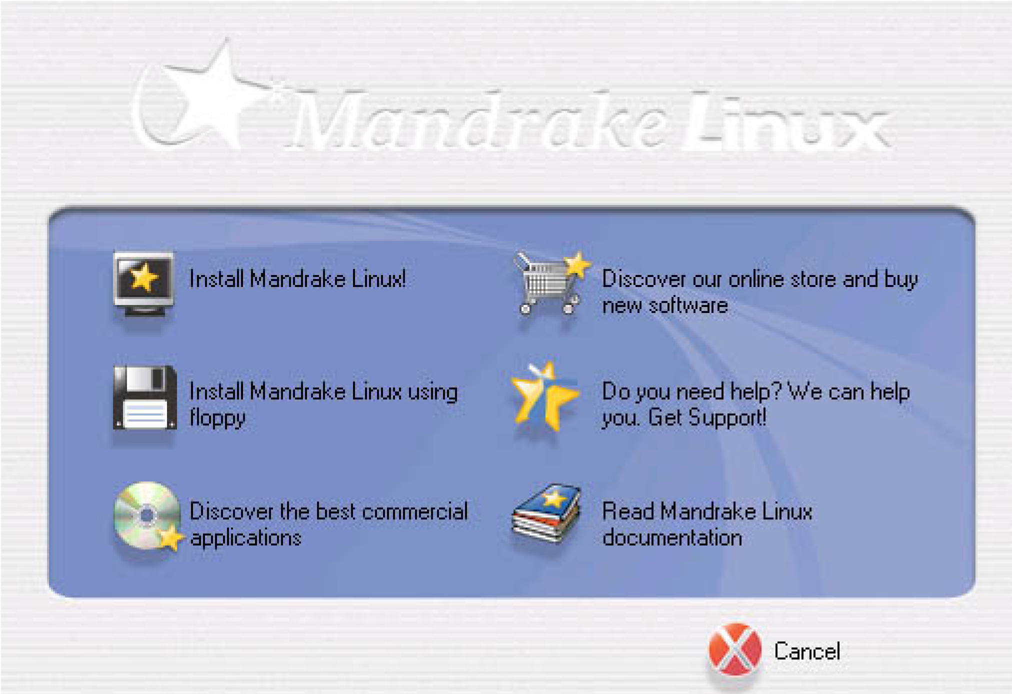
23. 安装完成！恭喜发财！

不经一番寒彻骨，焉得梅花扑鼻香，呵呵！您已经经过了『一番寒彻骨』了，恭喜您即将进入『扑鼻香』的境界！恭喜您，安装成功，请按下『确定』，然后取出光盘片，OK！等待登入吧！



24. 安装后的注意事项：

- 你可能会觉得很奇怪，为什么你的安装过程会跟我的不一样？！呵呵！因为每个人选择的套件都不尽相同，因此，如果你在安装的过程中选择了跟我不一样的套件，不用担心，安装过程会有些许的不相同是正常的！
- 在安装完成之后，请千万记得『取出光盘片』，不然又会在进入一次安装画面喔！
- 同时建议，安装完成之后，请进入您的 BIOS 当中，将开机的顺序改回来『 C、A 』或『 C only 』反正就是让硬盘开机啦！这样比较安全一些！
- 如果真的没有办法在开机的时候加载 Linux 的核心，以进行安装时，不要太担心，直接将 Mandrake 9.0 第一片可开机片放入 Windows 的系统当中，会出现下面图示：



再直接按下 Install Mandrake Linux! 即可安装啰！ ^_^y

好了！这样应该就已经安装完毕了！请继续往下看看吧！而且，相当的建议您，在正式的进行架站之前，请依序看一下底下的网页，最好不要跳着看，不然的话，嘿嘿嘿嘿！出现什么问题可不要怪我！因为，照着顺序看会对你的 Linux 认识比较有帮助啦！

多重开机安装流程与技巧

很多的朋友，包括我自己，由于工作的需要，常常需要两部不同的操作系统来处理日常生活与工作的杂事！那么我是否需要两部计算机呢？并不需要，只要一部计算机使用多重开机的方式来进行安装，嘿嘿！这样就 OK 啊！理论上是如此，不过实际上还需要一些小技巧呢！

- 硬盘重新规划的多重开机系统：
  如果你想要在你的 Linux 机器上同时安装 Windows ？可行吗？当然可行啰！况且目前很多的朋友手边只有一部计算机，但是又想要同时学习一下 Linux ，呵呵！那么安装多重操作系统实在是必须要的！好了！那要如何安装呢？以我前一阵子帮一个朋友规划的 Win98, Win2000, Linux 为例，我先将硬盘以 spfdisk 切割成两个 FAT partition，分别是 2GB 与 3GB ，预计安装 Win98 与 Win2000 （分别是 C: 与 D: ），然后再以 CD

开机后，分割最后的磁盘成为 / 与 Swap 两个！好了！如何安装：

1. 先以 Spfdisk 分割硬盘：由于 Windows 的 Fdisk 实在太慢了，我蛮喜欢使用 spfdisk 这个全中文的磁盘分割接口的!简单又方便!将硬盘切割成 C:2GB,D:3GB 即可！详细的 Spfdisk 执行范例可以到网络上搜寻一下教学文章吧！

2. 先安装 Win98 ：这个简单吧！用 98 开机片开机之后，直接安装，并且选择安装在 C 槽即可！

3. 再安装 Win2000：进入 Win98 之后，将 Win2000 的光盘片放进光驱中，屏幕会自动的跑出一个窗口，问你要不要升级，选择『是』，然后会进行一些小动作！在安装程序问到『升级安装或全新安装』的时候，请千万选择『全新安装』这个项目，并且不要升级硬盘扇区！然后在出现一个『问你安装目录所在』的问题时，进入选项里面，选择『要我自己挑选硬盘分割区』那个项目！然后接下来一直按下『确定』或『是』即可！之后，计算机会重新开机，开机完成之后会进入 Win2000 的安装画面，然后在出现『安装扇区』的时候，请选择 D 槽，并且选择『不要更改扇区档案系统』即可！接下来就会完成一些程序啦！

4. 最后才安装 Linux distribution ：是的，最后才安装 Linux ！安装的过程就如同上面提的，只不过在硬盘分割的地方会比较不一样就是了！！

5. 以 Lilo 或 grub 设定多重开机：是的，您必须选用 lilo 或 grub 来将您的开机程序设定一下，这个动作我们会在后头再谈，或者您可以在了解 vi 之后，直接翻到第十九章去瞧一瞧去！

- 在既存的 Windows 系统中加装 Linux 系统：
  另外再提供一个之前也曾经安装过的一个经验！恩！你可能会觉得奇怪，这个方法跟上一个方法有什么不同！？呵呵呵呵！最大的不同在于：

  我既存的 Windows 系统中的数据不想丢掉，并且我也没有新的硬盘来暂存我的系统或者是备份数据！假设原本我的 20 GB 硬盘中分割成 10GB, 10GB 两槽，但是我还想要安装 Linux ，且是在『旧系统仍然可以存活』的情况下！那该如何是好？！

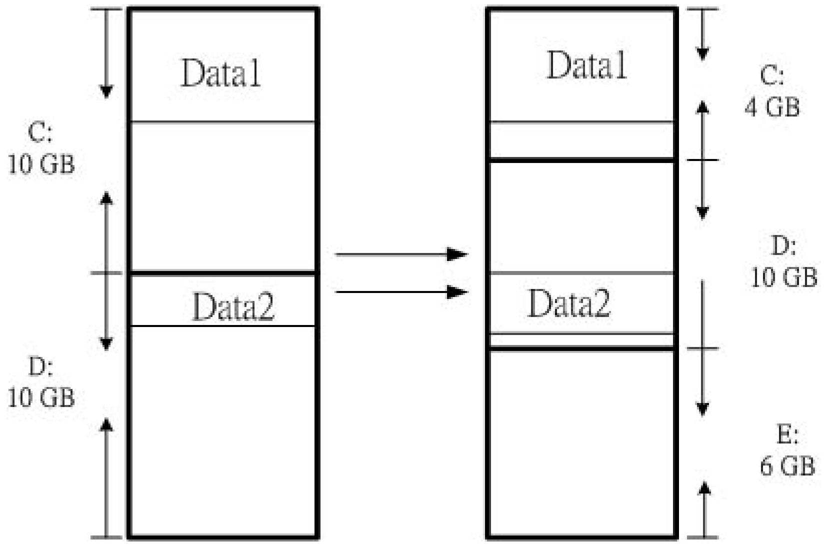
  这真的是很有趣的问题！早先在 Windows 系统中， VBird 就犯了一个错！C 槽给的太大了！基本上，系统文件不需要太大啦！通常我都喜欢 C 槽只给大约 4 GB 左右的空间（甚至更小），这是因为 C 槽是很需要备份的！如果太大的话，备份很麻烦！所以系统重置就会很花时间（因为所有的东西都要重新安装！我哩咧....！）！因此，我都习惯将 C 槽只给一点点的空间，然后再安装完并设定完所有的系统之后，马上以 Ghost 来备份我的系统！而所有的备份数据文件都摆放在 D 槽！此外，我的 OutLook Express 的书信目录也都不是摆在 C 槽!呵呵所以我不会很害怕 C 槽挂掉,因为,直接以 Ghost 还原即可啰！系统还原还不需要 30 分钟呢！

  这里就发生一个问题啦，假如原本的系统是 10GB, 10GB 的两槽，不过全部的有用到的资料量只有 10GB 不到！也就是还有空间来安装 Linux ，但是由于硬盘切割的不好，所

以伤脑筋！此外，我的原系统希望留下来，而且也希望可以安装 Linux ，要怎办？！我曾经这样做过：

- 由于 FAT 的扇区使用，其实只是在磁头区域（所谓的硬盘第零轨）规划而已，所以，我就将我的数据先以『磁盘重组』的方式将数据都归结在一起；

- 然后以 Spfdisk 将该硬盘的 FAT 表进行分割，注意喔！只是分割 FAT 表，并没有 format 喔！不过这里的技术性很高，需要特别注意！因为你是将 FAT 表重新划分，所以你的数据必须要在同一个扇区内！好了，我就将原本的 10GB 10GB 切割成 4GB、10GB 与 6GB 三槽！而且在 spfdisk 的帮助之下，顺利的在没有任何数据遗失的状况下，将我的硬盘由原先的两槽分割成三槽啰！那么一来，我就可以在我原本的 D 槽里面安装 Linux 啦！方法有点像底下的图示：



很神奇吧！数据还是在原来的地方，不过扇区的定位点改变了，还多出一个扇区！不过，这里要提醒大家，虽然 VBird 曾经以这个方法成功的将硬盘数据在不毁损的情况下，顺利的将硬盘切割完毕，不过，这个方法本身还是具有相当程度的风险，呵呵！因此不是很建议您这样做！尤其是当你的数据还很重要的时候！切记切记！

课后练习(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- Linux 的目录配置以『树状目录』来配置，至于磁盘分割区 (partition) 则需要与树状目录相配合！请问，在预设的情况下，在安装的时候系统会要求你一定要分割出来的两个 Partition 为何？
  答：

  就是根目录『/』与虚拟内存『Swap』

- 什么是 IDE 界面，一般而言，普通 PC 允许几个 IDE 界面与装置？
  答：

  IDE 为用来传输硬盘数据的一个汇流界面；
  共有 IDE1, IDE2 ，分别有 master 与 slave 所以共四个 IDE 装置支持！

- IDE2 的 master 之第一个 logical 磁盘中，其装置代号（文件名称）为何？
  答：

  /dev/hdc5

- 在硬盘分割 (Partition)时，最多有几个 primary + extended ？
  答：

  Primary + Extended 共四个，其中 Extended 只有一个！（ 更详细的硬盘与 MBR 可以参考 这里 这篇讨论 ）

- 若在分割的时候，在 IDE1 的 slave 硬盘中，分割『六个有用』的扇区（具有 filesystem 的），此外，有两个 primary 的扇区！请问六个扇区的代号？
  答：

  /dev/hdb1(primary)
  /dev/hdb2(primary)
  /dev/hdb3(extended)
  /dev/hdb5(logical 底下皆为 logical)
  /dev/hdb6
  /dev/hdb7
  /dev/hdb8
  请注意，5-8 这四个 logical 相加的总和为 3！

- 一般而言，在 RAM 为 64MB 或 128 MB 的系统中， swap 要开多大？
  答：

  Swap 可以简单的想成是虚拟内存，通常他的建议大小为 RAM 的两倍，但是实际上还是得视您的主机规格配备与用途而定。约两倍的 RAM ，亦即为 128 MB 或 256 MB ，可获得较佳效能！

- 什么是 GMT 时间？台北时间差几个钟头？
  答：

  GMT 时间指的是格林威治时间，为标准的时间，而台北时间较 GMT 快了 8 小时！

- Tap, SCSI 硬盘, RAID, printer 的装置代号？
  答：

Tap   : /dev/ht0 (IDE), /dev/st0 (SCSI);
SCSI H.D.: /dev/sd[a-p],
RAID   : /dev/md[0-15];
printer : /dev/lp[0-2]

- 如果我的磁盘分割时，设定了四个 Primary 扇区，但是磁盘还有空间，请问我还能不能使用这些空间？
  答：

  不行！因为最多只有四个 Primary 的磁盘分割槽，没有多的可以进行分割了！且由于没有 Extended ，所以自然不能再使用 Logical 分割说

- 我的 Mandrake 9.0 在安装的时候，进行 X-Window 的测试时都不会成功，要怎么办呢？
  答： X-Window System 的！万一还是没有办法登入 X-Window 的话，没有关系！不要害怕！等到后来『系统管理员篇』的时候，我们再来`入的谈一谈 X-Window 的设定吧！！ ^_^。而，如果万一不幸不小心按下了测试，要怎么办呢？屏幕已经一片漆黑了!@_@,没关系,此时可以按下 [Ctrl] + [Alt] + [F1] 就可以回到原先的画面啦！

- 通常在安装 Linux 的时候，最重要的就是磁盘分割了！请问：磁盘分割通常要分成几个步骤？
  答：

  1. 进行磁盘分割 partition ；
  2. 进行格式化 format ；

- 磁盘分割之后会有所谓的 Primary, Extended 与 Logical 的磁盘分割槽，请问何者为可使用的 Partition ？
  答：

  只有 Primary 与 Logical 为可用， Extended 为不可直接使用的 Partition ，还需要再次的分割成为 Logical 之后，才可以继续使用！而最大可分割出来的 Partition 应该有 64 个才对！

开机关机、在线求助与指令下达方式

最近更新日期：2005/08/03

由于 Linux 在运作的过程中，会有很多的程序常驻在内存中来执行，此外， 由于 Linux 的磁盘使用效能比较高，利用了异步的磁盘/内存数据传输的模式， 因此， Linux 系统是很怕不正常开关机的！因为，不正常开关机的结果， 将可能造成磁盘数据的损毁啊！(其实各个操作系统都很怕这个问题！)。 所以，在这个章节，鸟哥会跟大家介绍一下 Linux 正常开关机的步骤， 以及初次进入 Linux 的您，可以如何来操作文字接口的指令呢！要注意啊！ 学习文字接口真的是一件很不错的事喔！ ^_^

首次登入系统

登入系统有这么难吗？呵呵！并不难，不过，虽然说是这样说，然而很多人第一次登入 Linux 的感觉都是『接下来我要干啥？』 如果是以图形接口登入的话，或许还有很多好玩的事物，但是，要是以文字接口登入的话， 面对着一片黑压压的屏幕，还真不晓得要干嘛呢！嗯！为了让大家更了解如何正确的使用 Linux ，正确的登入与离开系统还是需要说明的！

首次登入 FC 图形接口

啊开机就开机呀！怎么还有所谓的登入与离开呀！？呵呵！开什么玩笑，在 Linux 里面，正确的开关机可是很重要的！因为， 不正常的关机可能会导致整个系统的扇区错乱，造成数据的毁损呢！这也是为什么通常我们的 Linux 主机都会加挂一个不断电系统啰！

在顺利的安装完成之后，就是要快乐的进入 Linux 的世界啦，这个时候，按下电源， 如果您预设是有启动图形接口的话，那么应该会出现如下的字样(这里是以 FC4 作为范例介绍的，各家版本多少会有点不同

的登入画面，这无所谓！ 只是作为选择操作系统的一个画面而已。不过要注意的是，开机前如果想要加入任何的参数， 就得在底下这个画面来增加喔！ ^_^）

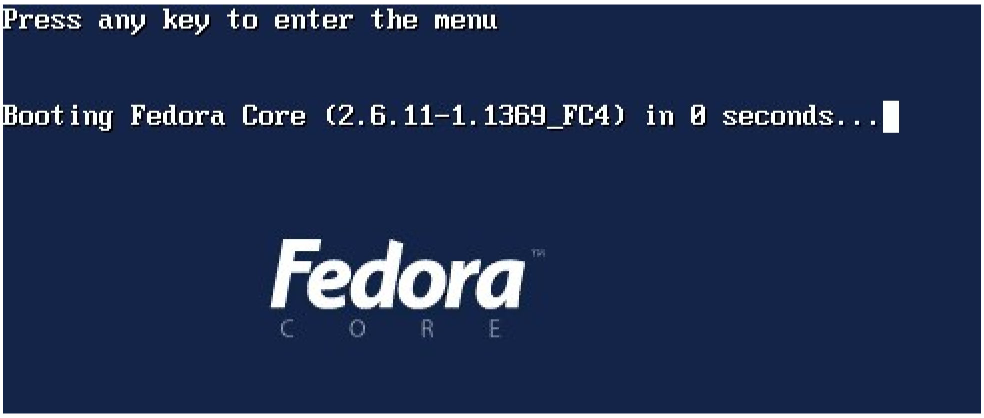


图 1 、 Linux 的 grub 开机选单画面

看到了吗？！这就是开机的选单啊！在上图的最上一行，告诉我们，如果想要进入额外的选单的话， 可以按任意键来进入，而这就是 grub 这个程序的功能了。另外，如果您在预设的时间内没有按下任何按键(在这一版的 Fedora ，预设是 3 秒钟啦！)，那么 grub 开机管理程序就会以系统预设的核心来开机。 事实上， grub 的功能还有很多，包含可以在系统发生错误的时候，以额外的参数来强制开机， 以顺利进行系统的修复等等功能呢！关于这个，我们留待后面的『系统管理员篇』再来详细的介绍这个玩意儿～

此外，如果是以另一个开机管理程序(loader)，也就是 LILO 来设定你 MBR 的开机选单时， 那么预设也是不会出现选单的！只会有出现『 boot: 』的画面而已，如果按下『 Enter 』 就会以预设的开机档来开机，如果按下『Tab』按键，就会出现其它可能的开机档啦！ 不过，在预设的情况下， FC4 并不会主动的安装 Lilo 呢！但比较旧的版本还是存在的啦！ 这部份我们同样在管理员篇再介绍！

假设我们是以 FC4 预设的开机核心来开机，那么接下来系统就会读取核心程序， 并且开始跑一些硬件搜寻的数据， 然后是一些服务的开启动作，就像下图一般：



图 2 、 开机过程的文字画面

如果一切都顺利，嘿嘿！就会进入图形画面了！如同底下的模样！(要注意， 我们这里预设是以图形接口

来登入 Linux 的，所以才会有这样的画面， 如果您是以纯文字接口来登入 Linux ，那么就不会有这些画面了。果真如此的话， 那么这部分您稍微瞧一瞧即可！）



图 3 、 开机过程的图形画面

如果在上图按下了『显示详细信息』时，就会显示出一些文字讯息了， 对于初学者来说，或许没有什么太大的意义，不过，还是先告知一下呢， 这个时候，您可以看一看，到底有多少程序已经被启动呢？

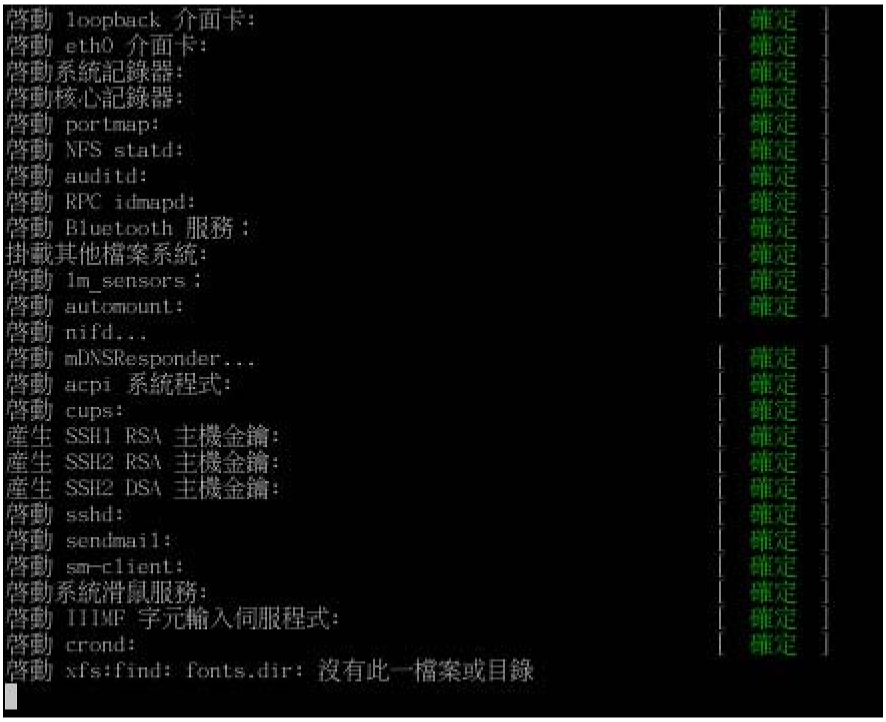


图 4 、 开机过程的图形画面

怕了吧？？有这么多不知名的咚咚已经在您的 Linux 里面启动了呢！ 里面其实有很多是我们不需要的，在未来您了解了 Linux 相关的知识之后， 就可以将那些不需要的程序(或称为服务)给他关掉了。目前还不需要紧张， 因为我们还没有连上 Internet 哪！还不需要太紧张啦！ ^_^

另外，如果您是使用本书介绍的 FC4 (Fedora Core IV)来安装您的 Linux ， 并且也安装了预设 X Window 启动的状态，那么您就得要设定 X Window 的使用环境啰！ 瞎密！？还需要设定 X Window 喔？没错啊！因为您可以修改时区啊、选择语系啊、设定屏幕相关的分辨率与色泽度啊等等的， 这些都是 X Window 环境下需要的咚咚呢，所以，当然得要设定一下啰。而且设定很简单啦！ 用鼠标点一点就好了，别紧张喔！^_^

---

1. 欢迎画面与授权

首先，屏幕会出现如下的欢迎画面，主要分为左右两个画面，左边仅是介绍流程到哪里， 右边才是内容设定的部分。至于最下方则是下个步骤与前个步骤的按钮。 我们按下『下一步』后，进入授权的说明。



图 5 、 X Window 设定的欢迎画面

在欢迎画面之后，会出现如下的授权声明，这个时候，当然给他『是的』就好了！



图 6 、 X Window 设定的授权同意书

---

2. 日期与时间的设定

接着下来，就是要设定目前的日期与时间了。您的系统时间可能会跑掉喔！ 所以，这个时候请调整回来吧！^_^ 画面左边可以点选正确的日期， 右方则可以填选正确的时间说！



图 7 、 X Window 设定的日期与时间

---

3. 分辨率与彩度的设定

接下来则是重头戏啦！就是 X Window 系统内的显示卡模块、屏幕分辨率与彩度设定。 一般来说，如果画面上显示的是您正确的显示卡，那么问题就不大了！而鸟哥喜好的分辨率， 大概就是 1024x768 那种大大的画面小小的字体～各人喜好啦！ 而如果您的显示卡内存没有很大的话，彩度可以调小一点！



图 8 、 X Window 设定的分辨率与彩度

---

4. 建立一般账户

一般来说，我们在操作 Linux 系统时，除非必要，否则不要使用 root 的权限， 这是因为管理员(root)的权限太大了！我们可能会随时不小心搞错了一个小咚咚， 结果却造成整个系统的挂点去.....所以，建立一个一般身份使用者来操作，才是好习惯。 举例来说，鸟哥都会建立一个一般身份使用者的账号(例如底下的 dmtsai)， 用这个账号来操作 Linux ，而当我的主机需要额外的 root 权限来管理时， 才使用身份转换指令(这个我们会在后面提到)来切换身份成为 root 来管理维护呢！ ^_^

如下图所示，登入的账号名称为 dmtsai ，而全名仅是一个简易的说明而已， 那个地方随便填没关系(不填也无所谓！)。但是两个密码栏均需填写， 他并不会显示出您打入的字符，而是以 * 取代。两个必须打相同的密码喔！



图 9 、 X Window 设定的建立一般账号使用者

---

5. 额外的音效与软件

如果您有声卡的话(或者是主机板内建的音效芯片)，就会出现如下的图示。 如果确定音效芯片名称没有问题，那就直接下一步即可。



图 10 、 X Window 设定的声卡确认

如果您还想要额外的安装其它的增强套件，这个时候可以在这里加入。 不过，我们使用预设的 FC4 ，不需要额外的 CD 来安装啦！



图 11 、 X Window 设定的额外的 CD

这样就完成了首次 X Window 的设定啰！



图 12 、 X Window 设定完成

接下来呢？嘿嘿！等着进入 X Window 的美美的画面吧！如下图所示，这是 Fedora 预设的登入画面。您可以看到中间请你输入『用户名称』，那就是账号啦！ 目前我们 Linux 系统上面应该会有一个 root 与刚刚才建立的 dmtsai 这两个账号呢。 你可以利用你刚刚自己建立的账号来登入喔！



图 13 、 等待登入的图形接口

另外，仔细看一下上图中的最下方还有四个选项呢，分别是『语言』、『作业阶段』、『重新开机』 与『关机』。他们各有什么功能呢？首先，我们先来看看语言有啥功能吧？ 用鼠标按下『语言』之后，就会出现如下的画面。嘿嘿！没错！您可以使用多种语言的显示呢！ 我们是使用繁体中文啊！

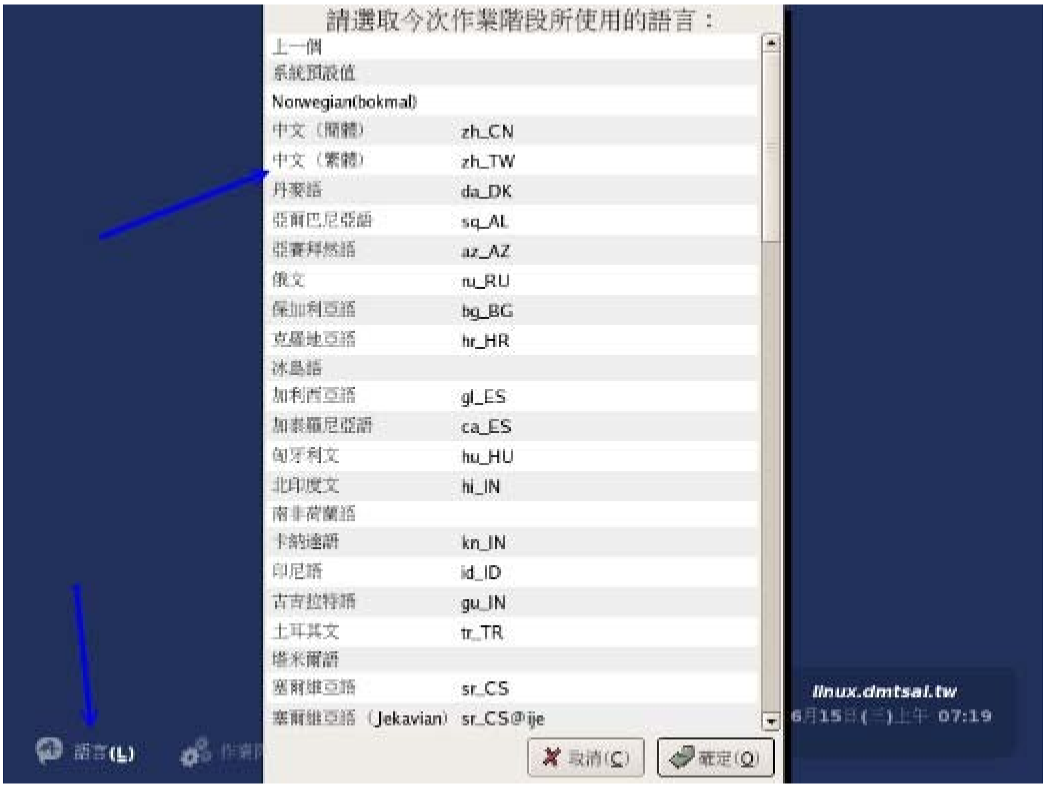


图 14 、 选择这次工作的语言

在接下来则是『作业阶段』，用鼠标按下他，出现如下画面。不要怀疑！ 在这里你就可以选择你喜欢的 X Window Manager 了！亦即是窗口管理员啊！ 因为鸟哥上次安装时，仅安装 KDE 而已，所以只会出现 KDE ，否则， 应该还有 GNOME 会出现在这个选单中喔！

Tips:

什么是 KDE 呢？为了让 X Window 的显示效果更佳，很多团体开始发展桌面应用的环境， KDE 就是其中一个。他们的目标就是发展出类似 Windows 桌面的一整套可以工作的桌面环境， KDE 是架构在 X Window 上面的，他可以进行窗口的定位、放大、缩小、 同时还提供很多的桌面应用软件，详情请参考 http://www.kde.org/。GNOME 则是另外一个计划！





图 15 、 选择喜好的 Window Manager

各位观众！接下来，哈哈！终于来到了登入的阶段了！如同下面所示， 输入账号后按下『Enter』再出现密码后，再输入密码，这里同样的， 屏幕上不会出现密码啦！会用星号(*)取代的喔！



图 16 、 登入时，账号的填入



图 17 、 登入时，密码的填入

接下来就能够进入 X Window 啦！不过，系统还会很好心的询问一下， 你要不要将这次的环境设定(KDE 啊、语言啊等等的)保留成为默认值呢？ 通常鸟哥都是选择默认值啦！所以就给他保留成默认值吧！



图 18 、 是否保留此次登入的设定

KDE 的简易操作

嘿嘿嘿嘿！真是的，历经千辛万苦，终于还是给我进入了 KDE 的画面喔～ 整个画面如下所示，主要分为两个区块，亦即上方深蓝色的桌面区， 以及下方的工作列(control panel)。在桌面上还有一些小图示(icons) 可以用来快速连结到某些内容。 其实跟微软的 Windows 桌面很像啦！那些小图示就是快捷方式啰！

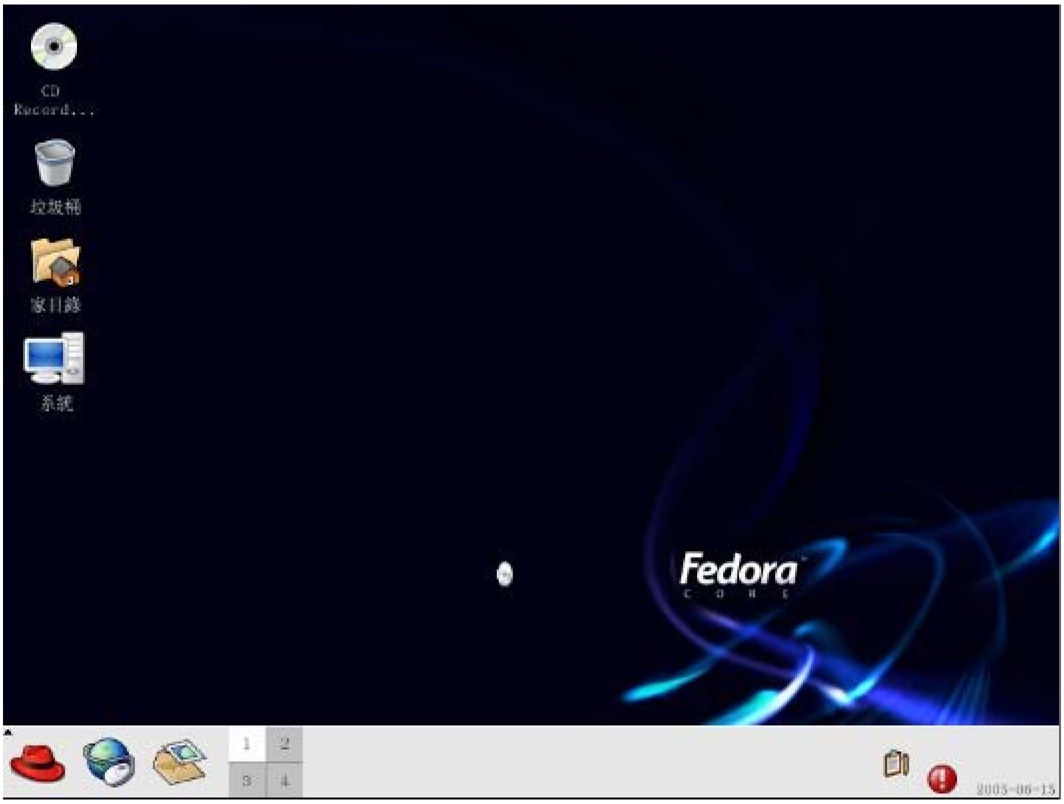


图 19 、 KDE 的桌面环境

整个桌面的使用方法几乎跟 Windows 一模一样，你可以在桌面上按下右键， 就可以有额外的选单出现；您也可以直接按下桌面上的『家目录』，就会出现类似 Windows 的 『档案总管』的档案/目录管理窗口，里面则出现您自己的工作目录；至于最下方的工作列， 最左边出现的三个小图示中，那个红色的帽子的功能(KDE menu)，就跟 Windows 的『开始』一样， 你按下红帽后，就会出现一个下拉式选单出来，您就可以选择其它的相关程序来执行了。

Tips:
最左边的图示中，其实在 KDE 原本的图示，是以一个大 K 来展现的。 Fedora 则修订成为小红帽。某些版本还是保留 KDE 的 K 图示呢！ 另外，关于家目录，记得我们之前说过 Linux 是多人多任务的操作系统吧？ 那么每个人自然应该都会有自己的『工作目录』，这个目录是使用者可以完全掌控的， 所以就称为『使用者个人家目录』了。一般来说，家目录都在 /home 底下， 以我的这次登入为例，我的账号是 dmtsai ，那么我的家目录就应该在 /home/dmtsai 啰！



至于工作列左边数来第二个 icon ，则是一个浏览器喔(Konqueror, 中文翻译为『征服家』) 。他是 KDE 发展的一个浏览器， 如果未来您可以连上 Internet 后，就可以利用这个浏览器来浏览网站了！而第三个 icon 则是信件收发软件，功能有点类似 Windows 的 outlook express 啰！ 总之，如果您用过 Windows 的话，嘿嘿！这个 KDE 的环境几乎与 Windows 相同， 你可以开始玩弄 X Window 啰。

且慢且慢！不知道您有没有发现在第三个 icon 的右边还有个四方形的咚咚， 里面还分成四个小方格，那是个什么咚咚？其实，他被称为虚拟桌面(Virtual Desktop)， 在你进入 KDE 后，应该是到『1』的画面，但是我们的 KDE 提供了四个虚拟桌面， 你还可以分别到其它三个桌面去瞧一瞧呢！这预设的四个桌面都

可以有自己的底图， 而如果你有很多窗口时，就可以放在不同的桌面中，还不会互相影响呢！ 赶紧去试看看吧！^_^

最后，在工作列的最右方有个小小的警告标志(惊叹号)，那是什么啊？！ 没有任何一个操作系统是绝对安全的！我们的 Linux 当然也是～ 而为了保持我们 Linux 的所有程序、套件的安全性，随时将套件更新到最新版本， 是一个很好的习惯啊！在 Fedora 当中，我们使用的是 Red Hat 发展的 up2date 更新方法，但是需要注册才能使用。由于我们还没有连上 Internet ，当然也就尚未注册资料， 所以这里才会显示一个惊叹号啦！无论如何，我们还可以透过其它较为快速的方法来升级， 不一定要用这个机制，所以，这个图形就先让他摆在这里，先不要理他。

Tips:
这里说不要理他，并不是不重要喔！设定系统的自动升级是目前所有知名的操作系统 都在努力进行的工作，我们 Fedora 可以利用社群提供的 yum 或 apt 等机制来更新， 不需要透过远在美国的 Fedora 计划的主机来更新啦！不过，因为这个图示判断是以 有没有注册来判定，所以，常常我们自己升级了，但这里还是会出现有问题的惊叹号～ 因此，鸟哥才说，先不要理他。另外，这么说您也就知道了，系统可能并不十分安全， 因此，在尚未提到更新方法之前，连上 Internet 可是很危险的喔！



至于更多的 X window 相关的使用技巧，以及相关的软件应用，鸟哥这里就不多说了， 因为鸟哥着重在 Linux 网络服务器的应用啊！ ^_^ 如果您还真的有兴趣， 建议您可以前往杨老师的网站上看看喔！http://apt.nc.hcc.edu.tw/docs/FC3_X/ 。

如果使用 KDE 完毕后，想要离开，那么按下工作列最左方的小红帽，选择最下方的『注销』， 会出现底下的画面，按下『End current session』就可以回到等待登入的画面啰。



图 20 、 注销 KDE

同时给他注意一下，如果不想要玩 Linux 了，想要关机时，务必按下那个『关机』的选项， 以出现如下画面后，选择关闭计算机，这样才行喔！不要直接按电源啊！拜托拜托！



图 21 、 关闭 Linux 主机

例题：(很简单的，请读者们自行操作找出答案喔)

- 如何在工作列 (Control panel) 上新增其它的 icons ？

- 尝试浏览一下 /etc 这个目录内，有哪些档案/目录存在？
- 请将 /etc/crontab 这个档案『复制』到您的家目录中；
- 尝试修改屏幕分辨率；
- 请尝试『搜寻』档案，档名为 crontab
- 在您按下桌面的『家目录』后，出现的窗口中，最右上角有个小钉子， 按下他之后，发生什么现象？
- 请修改四个 Virtual Desktop 的底色图案，让他们都不相同；
- 工作列的最右方原本是数字形态的时钟，请将他改为图形显示的时钟；
- 如何叫出控制台？控制台的『区域性』里面的『键盘对应』有何用处？

特殊小技巧：
一般来说，我们是可以手动来直接修改 X Window 的设定档的，不过， 修改完成之后 X Window 并不会立刻加载，必须要重新启动 X 才行 (特别注意，不是重新开机，而是重新启动 X ！）。那么如何重新启动 X 呢？ 最简单的方法就是在 X 的画面中直接按下 [Alt] + [Ctrl] + [Backspace]， 亦即是退格键，这样就可以直接重新启动 X 啰！也就可以直接读入设定档啰！ 另外，如果您的 X Window 因为不明原因导致有点问题时， 也可以利用这个方法来重新启动 X 喔！ ^_^

X window 与文字模式的切换

我们前面一直谈到的是 X Window 的 KDE 环境，那么在这个环境里面有没有纯文字接口的环境啊？ 当然有啊！但是，要怎么切换 X Window 与文字模式呢？注意喔，通常我们也称文字模式为 终端机接口, terminal 或 console 喔！ Linux 预设的情况下， 会提供六个 Terminal 来让使用者登入，切换的方式为使用：[Ctrl] + [Alt] + [F1]~[F6] 的组合按钮。

同时，系统为了判断，会将 [F1] ~ [F6] 定义为 tty1 ~ tty6 的操作接口环境。也就是说，当您按下 [crtl] + [Alt] + [F1] 这三个组合按钮时，就会进入到 tty1 的 terminal 界面中了。同样的 [F2] 就是 tty2 啰！那么如何回到刚刚的 KDE 呢？ 很简单啊！按下 [Ctrl] + [Alt] + [F7] 就可以了！

Tips:
注：某些 Linux distribution 会使用到 F8 这个终端接口做为他的桌面终端机，例如 OpenLinux Server 3.1.1 ，所以这部份还不是很统一！无论如何，尝试按按 F7 or F8 就可以知道啰！



- [Ctrl] + [Alt] + [F1] ~ [F6] ：文字接口登入 tty1 ~ tty6 终端机；
- [Ctrl] + [Alt] + [F7] ：图形接口桌面。

这也就是说，如果您是以文字接口登入的话，那么您可以有 tty1 ~ tty6 这六个文字接口的终端机玩，但是图形接口 则没有任何东西。 至于以图形接口登入的话，就可以使用图形接口跟文字接口啰！ 而如果您是以文字接口启动 Linux 的，也就是说，您的 tty7 预设是没有东西的， 那您可以直接下达：

```
[root@linux ~]# startx
```

『理论上』就可以启动图形接口啦！当然，『前提是您的 X Window 需要设定 OK，且您有安装 KDE/GNOME 等桌面系统才行』。 好啦，我们知道在 Linux 开机之后，可以进入 X Window 或者是纯文字接口环境， 那么这两种环境是否可以变更呢？呵呵！那就涉及所谓的『Run Level』了！ 你可以将预设启动的 X Window (Run level 等级为 5)改为不启动(Run level 3)， 只要修订一下 /etc/inittab 这个档案的内容，就能

够决定呢！ 因为我们尚未提到 vi 以及开机过程的详细信息，所以啊， 这部分得到系统管理员篇幅的时候再说明！别担心，再仔细的看下去吧！

以文字模式登入 linux

好了，刚刚我们有提到按下 [Ctrl] + [Alt] + [F1] 可以来到 tty1 的 terminal 当中， 而如果您是使用纯文字接口 (其实是 run level 3) 启动 Linux 主机的话， 那么预设就是会来到 tty1 这个环境中。这个环境的等待登入的画面有点像这样：

```
Fedora Core release 3 (Heidelberg)
Kernel 2.6.11-1.27_FC3 on an i686

linux login: root
Password:
[root@linux ~]# _
```

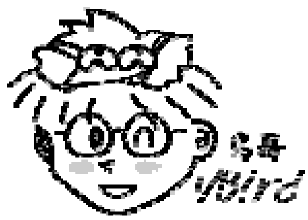
上面显示的内容是这样的：

- 第一行显示的是您的 Linux distribution 与版本；
- 第二行显示的是您的 Linux 核心版本 (2.6.11-1.27_FC3) ，以及您的硬件等级 (i686)。
- 第三行显是您的主机名称 (linux) ，至于 login 后面则是需要你输入登入者的账号。 在这里请输入您想要登入的使用者账号。我们直接以 root 来登入。注意， 那个 root 就是『系统管理员』，也就是『超级使用者, Super User』， 在 Linux 主机之内，这个账号代表的是『无穷的权力！』，任何事都可以进行的， 因此，使用这个账号要『粉小心！』
- 第四行则在第三行输入后才会出现，要你输入密码啰！请注意，在输入密码的时候， 屏幕上面『不会显示任何的字样！』， 所以不要以为你的键盘坏掉去！
- 第五行则是正确登入之后才显示的讯息，最左边的 root 显示的是『目前使用者的账号』， 而 @ 之后接的 linux 则是『主机名称』，至于最右边的 ~ 则指的是 『目前所在的目录』，那么那个 # 则是我们常常讲的『提示字符』啦！

Tips:
那个 ~ 符号代表的是『使用者的家目录』的意思，他是个『变量！』这相关的意义我们会在后续的章节依序介绍到。举例来说， root 的家目录在 /root， 所以 ~ 就代表 /root 的意思～而 dmtsai 的家目录在 /home/dmtsai， 所以如果您以 dmtsai 登入时，他看到的 ~ 就会等于 /home/dmtsai 喔！

至于提示字符方面，在 Linux 当中，预设 root 的提示字符为 # ，而一般身份使用者的提示字符为 $ 。

还有，上面的第一、第二行的内容其实是来自于 /etc/issue 这个档案喔！



好了这样就是登入主机了！很快乐吧！耶～

另外，在上面的例子当中，鸟哥是以 root 这个系统管理员身份的账号来登入的。 但是，在一般时刻的

Linux 使用情况中，为了『系统与网络安全』的考虑，通常我们都希望大家不要以 root 身份来登入主机的。这是因为系统管理员账号 root 具有无穷大的权力， 例如他可以删除任何一个档案或目录，因此，若您以 root 身份登入 Linux 系统，一个不小心下错指令， 这个时候可不是『欲哭无泪』就能够解决的了问题的～因此，一个称职的网络/系统管理人员， 通常都会具有两个账号，平时以自己的一般账号来使用 Linux 主机的任何资源， 有需要动用到系统功能修订时，才会转换身份成为 root 呢！所以， 鸟哥强烈建议您建立一个普通的账号来供自己平时使用喔！ 更详细的账号讯息，我们会在后续的『账号管理』章节中再次提及！ 这里先有概念即可！

那么如何离开系统呢？其实应该说『注销 Linux』才对！注销很简单，直接这样做：

```
[root@linux ~]# exit
```

就能够注销 Linux 了。但是请注意：『离开系统并不是关机！』 基本上，Linux 本身已经有相当多的工作在进行，您的登入也仅是其中的一个『工作』而已， 所以当您离开时，那么该工作就停止了，不过其它的工作但此时 Linux 其它的工作是还是进行的！在后面我们再来提如何正确的关机，这里先建立起这个概念即可！

文字模式下指令的下达

其实我们所谓的『文字模式』就是指你在登入 Linux 的时候，得到的一个 Shell 啦！ 那么什么是 Shell 呢？关于这个 Linux 重要的 bash Shell 的作用我们会在后面提到， 这里您先有个概念就好了。 Shell 提供我们使用者一些工具，可以透过这个工具， 来控制 kernel 的动作啰！ ^_^。好吧！开始来练一练打字了先！

开始下达指令

其实整个指令下达的方式很简单，您只要记得几个重要的概念就可以了。 举例来说，你可以这样下达指令的：

```
[root@linux ~]# command [-options] parameter1 parameter2 ...
                    指令      选项       参数(1)     参数(2)
说明：
0. 一行指令中第一个输入的绝对是『指令(command)』或『可执行档案』
1. command 为指令的名称，例如变换路径的指令为 cd 等等;
2. 中刮号[]并不存在于实际的指令中，而加入参数设定时，通常为 - 号，例如 -h;
   有时候完整参数名称会输入 -- 符号，例如 --help;
3. parameter1 parameter2.. 为依附在 option 后面的参数，
   或者是 command 的参数;
4. command, -options, parameter1.. 这几个咚咚中间以空格来区分，
   不论空几格 shell 都视为一格;
5. 按下 [Enter] 按键后，该指令就立即执行。[Enter] 按键为 <CR> 字符，
   他代表着一行指令的开始启动。
6. 指令太长的时候，可以使用 \ 符号来跳脱 [Enter] 符号，
   使指令连续到下一行。注意！ \ 后就立刻接特殊字符。
其它：
```

```
a. 在 Linux 系统中，英文大小写字母是不一样的。举例来说， cd 与 CD 并不同。
b. 更多的介绍等到 bash 时，再来详述。
```

注意到上面的说明当中，『第一个被输入的数据绝对是指令或者是可执行的档案』！ 这个是很重要的概念喔！还有，按下 [Enter] 键表示要开始执行此一命令的意思。来，我们实际操作： 以 ls 这个『指令』列出『/root』这个目录下的『所有隐藏文件与相关的档案属性』， 档案的属性的 option 为 -al ，所以：

```
[root@linux ~]# ls -al /root
[root@linux ~]# ls          -al   /root
```

上面这两个指令的下达方式是一模一样的执行结果喔！为什么？请参考上面的说明吧！ 关于更详细的文字模式使用方式，我们会在『Shell 与 Shell Scripts』篇幅中再来强调喔！ 此外，请特别留意，在 Linux 的环境中， 『大小写字母是不一样的东西！』也就是说，在 Linux 底下， VBird 与 vbird 这两个档案是『完全不一样的』档案呢！所以， 您在下达指令的时候千万要注意到您的指令是大写还是小写。例如当您输入底下这个指令的时候，看看有什么现象：

```
[root@linux ~]# date
[root@linux ~]# Date
[root@linux ~]# DATE
```

很好玩吧！不一样的大小写显示的结果会有错误讯息发生呢！ 因此，请千万记得这个状态呦！好啰，底下我们来练习一下一些简单的指令，好让您可以了解指令下达方式的模式：

另外，很多时候您会发现，咦！怎么我输入指令之后出现的是乱码？？ 这跟鸟哥说的不一样啊！呵呵！不要紧张～我们前面提到过， Linux 是支持多国语系的，若可能的话， 屏幕的讯息是会以该支持语系来输出的。但是，我们的终端机接口 (terminal) 在预设的情况下， 无法支持以中文编码输出数据的。这个时候，我们就得将支持语系改为英文，才能够显示出正确的讯息。 那怎么做呢？您可以这样做：

```
[root@linux ~]# LANG=en
[root@linux ~]# LANGUAGE=en
[root@linux ~]# LC_ALL=en
[root@linux ~]# LC_CTYPE=en
[root@linux ~]# LC_TIME=en
```

注意一下，上面每一行指令都是用等号『 = 』连接并且等号两边没有空格喔！是连续输入的！ 这样一来，就能够在『这次的登入』察看英文讯息啰！为什么说是『这次的登入』呢？ 因为，如果您注销 Linux 后，刚刚下达的指令就没有用啦！ ^_^，这个我们会在 bash shell 章节中好好聊一聊的！

基础指令的操作

底下我们立刻来操作几个简单的指令看看啰！

- 显示日期的指令： date
- 显示日历的指令： cal
- 简单好用的计算器： bc

---

1. 显示日期的指令： date

如果在文字接口上面，想要知道目前的时间，那么就直接在指令列模式输入 date 即可显示：

```
[root@linux ~]# date
Thu Jun 23 11:32:02 CST 2005
```

上面是显示：星期四, 六月二十三日, 11:32 分， 02 秒，在 2005 年的 CST 时区！请赶快动手做做看呦！好了，那么如果我想要让这个程序显示出『 2005/06/23 』这样的日期显示方式呢？ 那么就使用 date 的相关功能吧！

```
[root@linux ~]# date +%Y/%m/%d
2005/06/23
[root@linux ~]# date +%H:%M
11:35
```

那个『 +%Y%m%d 』就是 date 的一些参数功能啦！很好玩吧！那你问我， 鸟哥怎么知道这些参数的啊？要背起来吗？当然不必啦！底下再告诉你！

---

2. 显示日历的指令： cal

那如果我想要列出目前这个月份的月历呢？呵呵！直接给他下达 cal 即可！

```
[root@linux ~]# cal
      June 2005
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
```

基本上， cal (calendar) 这个指令可以做的事情还很多，你可以显示整年的月历情况：

```
[root@linux ~]# cal 2005
                               2005

       January               February                 March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                   1          1  2  3  4  5          1  2  3  4  5
 2  3  4  5  6  7  8    6  7  8  9 10 11 12    6  7  8  9 10 11 12
 9 10 11 12 13 14 15   13 14 15 16 17 18 19   13 14 15 16 17 18 19
16 17 18 19 20 21 22   20 21 22 23 24 25 26   20 21 22 23 24 25 26
23 24 25 26 27 28 29   27 28                  27 28 29 30 31
30 31
        April                   May                   June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2    1  2  3  4  5  6  7             1  2  3  4
 3  4  5  6  7  8  9    8  9 10 11 12 13 14    5  6  7  8  9 10 11
10 11 12 13 14 15 16   15 16 17 18 19 20 21   12 13 14 15 16 17 18
17 18 19 20 21 22 23   22 23 24 25 26 27 28   19 20 21 22 23 24 25
```

```
24 25 26 27 28 29 30   29 30 31               26 27 28 29 30

        July                  August               September
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2       1  2  3  4  5  6                1  2  3
 3  4  5  6  7  8  9    7  8  9 10 11 12 13    4  5  6  7  8  9 10
10 11 12 13 14 15 16   14 15 16 17 18 19 20   11 12 13 14 15 16 17
17 18 19 20 21 22 23   21 22 23 24 25 26 27   18 19 20 21 22 23 24
24 25 26 27 28 29 30   28 29 30 31            25 26 27 28 29 30
31
       October               November               December
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                   1          1  2  3  4  5                1  2  3
 2  3  4  5  6  7  8    6  7  8  9 10 11 12    4  5  6  7  8  9 10
 9 10 11 12 13 14 15   13 14 15 16 17 18 19   11 12 13 14 15 16 17
16 17 18 19 20 21 22   20 21 22 23 24 25 26   18 19 20 21 22 23 24
23 24 25 26 27 28 29   27 28 29 30            25 26 27 28 29 30 31
30 31
```

也就是说，基本上， cal 接的语法为：

```
[root@linux ~]# cal [month] [year]
```

所以，我想要知道 2005 年 7 月的月历，可以直接下达：

```
[root@linux ~]# cal 7 2005
      July 2005
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

所以，未来您可以很轻易的就以 cal 来取得日历上面的日期啰！简直就是万年历啦！ ^_^

---

3. 简单好用的计算器： bc

如果我想要使用简单的计算器呢？很容易呀！就使用 bc 即可！在输入 bc 之后， 显示出版本信息之后，就进入到等待指示的阶段。如下所示：

```
[root@linux ~]# bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
_<==这个时候，光标会停留在这里等待您的输入
```

事实上，我们是『进入到 bc 这个指令的工作环境当中』了！ 就好像我们在 Windows 里面使用『小算盘』一样！所以，我们底下尝试输入的数据， 都是在 bc 程序当中在进行运算的动作。所以啰，您输入的数据当然就得要符合 bc 的要求才行！ 在基本的 bc 计算器操作之前，先告知几个使用的运算子好了：

- + 加法
- - 减法
- * 乘法
- / 除法
- ^ 指数
- % 余数

好！让我们来使用 bc 计算一些咚咚吧！

```
[root@linux ~]# bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
1+2+3+4  <==只有加法时
10
7-8+3
2
10*52
520
10%3     <==计算『余数』
1
10^2
100
10/100   <==这个最奇怪！不是应该是 0.1 吗？
0
quit     <==离开 bc 这个计算器
```

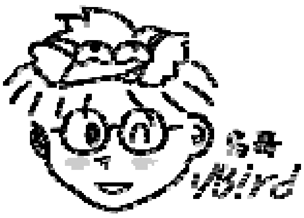
在上表当中，粗体字表示输入的数据，而在每个粗体字的底下就是输出的结果。 咦！每个计算都还算正确，怎么 10/100 会变成 0 呢？这是 因为 bc 预设仅输出整数，如果要输出小数点下位数，那么就必须要执行 scale=number ，那个 number 就是小数点位数，例如：

```
[root@linux ~]# bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
scale=3     <==没错！就是这里！！
1/3
.333
340/2349
.144
```

```
quit
```

好了！就是这样子啦！简单的很吧！以后你可以轻轻松松的进行加减乘除啦！

> Tips:
> 如果照前面说的，我们执行 bc 会进入 bc 的软件功能， 那么我怎么知道目前等待输入的地方是某个软件的功能还是 shell 的可输入指令的环境下？ 其实，在你进入 Linux 的时候，就会出现提示字符了不是吗？以我们上头的例子来说， 提示字符就是『 [root@linux ~]# 』，如果你发现在你等待输入的地方并非提示字符， 那通常就是已经进入到某个软件的功能当中啦！要注意喔！



重要的几个热键[Tab], [ctrl]-c, [ctrl]-d

在继续后面的章节之前，这里很需要跟大家再来报告一件事，那就是我们的文字模式里头具有很多的功能按键， 这些按键可以辅助我们进行指令的编写与程序的中断呢！这几个按键请大家务必要记住的！很重要喔！

- [Tab] 按键

[Tab] 按键就是在键盘的大写灯切换按键([Caps Lock])上面的那个按键！ 在各种 Unix-Like 的 Shell 当中，这个 [Tab] 按键算是 Linux 的 Bash shell 最棒的功能之一了！ 他具有『命令补全』与『档案补齐』的功能喔！可以让我们少打很多字，但重点是， 可以避免我们打错指令或文件名称呢！很棒吧！但是 [Tab] 按键在不同的地方输入，会有不一样的结果喔！ 我们举下面的例子来说明。上一小节我们不是提到 cal 这个指令吗？如果我在指令列输入 ca 再按两次 [tab] 按键，会出现什么讯息？

```
[root@linux ~]# ca[tab][tab]    <==[tab]按键是紧接在 a 字母后面！
# 上面的 [tab] 指的是『按下那个 tab 键』，不是要您输入 ca[... 的意思喔！
cadaver             callgrind_control   capiinit            case
cal                 cancel              capinfos            cat
calibrate_ppa       cancel.cups         captoinfo           catchsegv
caller              capifax             card
callgrind           capifaxrcvd         cardctl
callgrind_annotate  capiinfo            cardmgr
```

发现什么事？所有以 ca 为开头的指令都被显示出来啦！很不错吧！ 那如果你输入 ls -al ~/.bash 两个 [tab] 会出现什么？

```
[root@linux ~]# ls -al ~/.bash[tab][tab]
.bash_history  .bash_logout   .bash_profile  .bashrc
```

咦！在该目录下面所有以 .bash 的文件名称都会被显示出来了呢！注意看上面两个例子喔， 我们按 [tab] 按键的地方如果是在 command (第一个输入的数据) 后面时，他就代表着 『命令补全』，如果是接在第二个字以后的，就会变成『档案补齐』的功能了！

- [Tab] 接在一串指令的第一个字的后面，则为命令补全；
- [Tab] 接在一串指令的第二个字以后时，则为『档案补齐』！

善用 [tab] 按键真的是个很好的习惯！可以让您避免掉很多输入错误的机会！！！

- [Ctrl]-c 按键

在 Linux 底下，如果您输入了错误的指令或参数， 有的时候这个指令或程序会在系统底下『跑不停』这个时候怎么办？别担心，如果您想让当前的程序『停掉』的话，可以输入：[Ctrl] 与 c 按键 (先按着 [Ctrl] 不放，且再按下 c 按键，是组合按键！)， 那就是 中断目前程序 的按键啦！ 举例来说，如果您输入了『 find / -type vbird 』这个指令时， 系统会开始跑一些东西 (先不要理会这个指令串的意义)，此时你给他按下 [Ctrl]-c 组合按键，嘿嘿！是否立刻发现这个指令串被终止了！就是这样的意思啦！

不过你应该要注意的是，这个组合键是可以将正在运作中的指令中断的， 如果您正在运作比较重要的指令，可别急着使用这个组合按键喔！ ^_^

- [Ctrl]-d 按键

那么 [Ctrl]-d 是什么呢？就是 [Ctrl] 与 d 按键的组合啊！这个组合按键通常代表着： 『键盘输入结束 (End Of File, EOF 或 End Of Input)』的意思！ 另外，他也可以用来取代 exit 的输入呢！例如您想要直接离开文字接口， 可以直接按下 [Ctrl]-d 就能够直接离开了(相当于输入 exit 啊！)。
总之，在 Linux 底下，文字接口的功能是很强悍的！要多多的学习他， 而要学习他的基础要诀就是......多使用、多熟悉啦！

错误讯息的察看

万一我下达了错误的指令怎么办？不要紧呀！您可以 藉由屏幕上面显示的错误讯息来了解你的问题点，那就很容易知道如何改善这个错误讯息啰！举个例子来说，假如想执行 date 却打错成为 DATE 时，这个错误的讯息是这样显示的：

```
[root@linux ~]# DATE
-bash: DATE: command not found
```

上面那个 bash: 表示的是我们的 Shell 的名称，那么什么是 Shell 呢？还记不记得我们在 什么是 Linux 的时候提到的『使用者、使用者接口、核心、硬件』的架构呢？呵呵！那个 shell 就是使用者接口啰！在 Linux 底下预设的使用者接口就是 bash shell 啰！

好了，那么上面的例子说明了， bash 有错误，什么错误呢？ bash告诉你：

DATE: command not found

字面上的意思是说『指令找不到』，那个指令呢？就是 DATE 这个指令啦！ 所以说，系统上面可能并没有 DATE 这个指令啰！就是这么简单！那如果是底下的样子呢？

```
[root@linux ~]# cal 13 2005
cal: illegal month value: use 1-12
```

呵呵！这下子换到 cal 警告你啦， illegal month value: use 1-12，看不懂英文？ 没关系，又不是考试，赶快拿本英文字典在旁边对照着看呀！意思是说『 不合法的月份值，应该使用 1-12 之间的数字！』所以各位看倌您看看，跟着屏幕的错误讯息瞧，很容易知道问题的错误是什么吧！ 因此，以后如果出现了问题，屏幕上的讯息真的是很重要的呢！不要忽略了他呦！
先介绍这几个指令让您玩一玩先，更详细的指令操作方法我们会在第三篇的时候再进行介绍！好了，万一我在操作 date 这个指令的时候，手边又没有这本书，我要怎么知道要如何加参数， 好让输出的结果符合我想要的输出格式呢？嘿嘿！到下一节鸟哥来告诉你怎么办吧！

Linux 系统上的在线求助 man page/info page

先来了解一下， Linux 有多少指令呢？在文字模式下，你可以直接按下两个 [Tab] 按键，看看总共有多少指令？？呵呵！少说也有 2000 多个以上的指令！！ 那在 Linux 里面到底要不要背『指令』啊？可以啊！你背啊！这种事，鸟哥这个『忘性』特佳的老人家， 实在是背不起来 @_@ ～当然啦，有的时候为了要考试（例如一些认证考试等等的）还是需要背一些重要的指令。 不过，鸟哥主要还是以理解『在什么情况下，应该要使用哪方面的指令』为准的！

既然不需要背指令，那么鸟哥如何知道每个指令的详细用法？还有，某些设定档的内容到底是什么？ 这个可不需要担心啊！因为在 Linux 开发的软件大多数都是自由软件，而这些软件的开发者为了让大家能够了解指令的用法， 都会自行制作很多的文件，而这些文件也可以直接在在线就能够轻易的被使用者查询出来喔！很不赖吧！ 这根本就是『在线说明文件』嘛！哈哈！没错！确实如此。

我们底下就来谈一谈，Linux 到底有多少的在线文件数据呢？

---

man page

嗄？不知道怎么使用 date 这个指令？嘿嘿嘿！不要担心，我们 Linux 上面的在线求助系统已经都帮您想好要怎么办了，所以你只要使用简单的方法去寻找一下说明的内容， 马上就清清楚楚的知道该指令的用法了！怎么看呢？就是找男人(man) 呀！？喔！不是啦！这个 man 是 manual（操作说明）的简写啦！只要下达： 『man date』 马上就会有清楚的说明出现在你面前喔！如下所示：

```
[root@linux ~]# LANG="en"
# 还记得这个咚咚的用意吧？前面提过了，是为了『语系』的需要啊！下达过一次即可！
[root@linux ~]# man date
DATE(1)                          User Commands                         DATE(1)

NAME
       date - print or set the system date and time

SYNOPSIS
       date [OPTION]... [+FORMAT]
       date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION
       Display the current time in the given FORMAT, or set the system date.

       -d, --date=STRING
              display time described by STRING, not 'now'

       -f, --file=DATEFILE
              like --date once for each line of DATEFILE
```

```
       -ITIMESPEC, --iso-8601[=TIMESPEC]
              output  date/time  in ISO 8601 format.  TIMESPEC='date' for date
              only, 'hours', 'minutes', or 'seconds' for date and time to  the
              indicated  precision.   --iso-8601  without TIMESPEC defaults to
              'date'.
................(略)....
AUTHOR
       Written by David MacKenzie.

REPORTING BUGS
       Report bugs to .

COPYRIGHT
       Copyright ?2004 Free Software Foundation, Inc.
       This is free software; see the source for copying conditions.  There is
       NO warranty; not even for MERCHANTABILITY or FITNESS FOR  A  PARTICULAR
       PURPOSE.

SEE ALSO
       The  full documentation for date is maintained as a Texinfo manual.  If
       the info and date programs are properly installed  at  your  site,  the
       command

              info coreutils date

       should give you access to the complete manual.

date (coreutils) 5.2.1            May 2005                         DATE(1)
```

看！马上就知道一大堆的用法了！如此一来，不就可以知道 date 的相关参数了吗？呵呵！真方便！ 而出现的这个屏幕画面，我们称呼他为 man page ， 您可以在里头查询他的用法与相关的参数说明。如果仔细一点来看这个 man page 的话， 您会发现几个有趣的东西。

首先，在上个表格的第一行，您可以看到的是：『DATE(1)』， DATE 我们知道是指令的名称， 那么 (1) 代表什么呢？他代表的是『一般使用者可使用的指令』的意思！咦！还有这个用意啊！！呵呵！ 没错～在查询数据的后面的数字是有意义的喔！他可以帮助我们了解或者是直接查询相关的资料。 常见的几个数字的意义是这样的：

代号	代表内容
1	使用者可以操作的指令或可执行文件
2	系统核心可呼叫的函数与工具等
3	一些常用的函数(function)与函式库(library)

4	装置档案的说明
5	设定档或者是某些档案的格式
6	游戏(games)
7	惯例与协议等，例如 Linux 标准档案系统、 网络协议、ASCII code 等等的说明内容
8	系统管理员可用的管理指令
9	跟 kernel 有关的文件

所以，未来您如果使用 man page 在察看某些数据时，就会知道该指令/档案所代表的基本意义是什么了。举例来说，如果您下达了 man null 时，会出现的第一行是：『NULL(4)』，对照一下上面的数字意义， 嘿嘿！原来 null 这个玩意儿竟然是一个『装置档案』呢！很容易了解了吧！？

再来，man page 的内容也分成好几个部分来加以介绍该指令呢！就是上头 man date 那个表格内， 以 NAME 作为开始介绍，最后还有个 SEE ALSO 来作为结束。基本上， man page 大致分成底下这几个部分：

代号	内容说明
NAME	简短的指令、数据名称说明
SYNOPSIS	简短的指令下达语法(syntax)简介
DESCRIPTION	较为完整的说明，这部分最好仔细看看！
OPTIONS	针对 SYNOPSIS 部分中，有列举的所有可用的参数说明
COMMANDS	当这个程序(软件)在执行的时候，可以在此程序(软件)中下达的指令
FILES	这个程序或数据所使用或参考或连结到的某些档案
SEE ALSO	可以参考的，跟这个指令或数据有相关的其它说明！
EXAMPLE	一些可以参考的范例
BUGS	是否有相关的臭虫！

有时候除了这些外，还可能会看到 Authors 与 Copyright 等等，不过也有很多时候仅有 NAME 与 DESCRIPTION 等部分。 通常鸟哥在查询某个数据时，一定会察看 NAME 约略看一下这个数据的意思，再详看一下 DESCRIPTION ，这个 DESCRIPTION 会提到很多相关的资料与使用时机，从这个地方可以学到很多小细节呢！而如果这个指令其实很熟悉了 (例如上面的 date)，那么鸟哥主要就是查询关于 OPTIONS 的部分了！可以知道每个参数的意思， 这样就可以下达比较细部的指令内容呢！最后，鸟哥会再看一下，啊跟这个资料有关的还有哪些东西可以使用的？ 举例来说，上面的 SEE ALSO 就告知我们还可以利用『info coreutils date』来进一步查阅资料， 某些说明内容还会列举有关的档案 (FILES 部分) 来提供我们参考！这些都是很有帮助的！

好了，大致上了解了 man page 的内容后，那么，在 man page 当中我还可以利用哪些按键来帮忙查阅呢？首先，如果要向下翻页的话，可以按下键盘的 空格键 ，也可以使用 [Page Up] 与 [Page Down] 来翻页呢！同时，如果您知道某些关键词的话， 那么可以在任何时候输入『 /word 』，来主动搜寻关键词！例如在上面的搜寻当中， 我输入了 /date 会变成怎样？

```
DATE(1)                          User Commands                          DATE(1)

NAME
       date - print or set the system date and time

SYNOPSIS
       date [OPTION]... [+FORMAT]
       date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION
       Display the current time in the given FORMAT, or set the system date.

...........(中间省略)........

/date
```

看到了吗？您按下『/』之后，光标应该就会移动到屏幕的最下面一行， 并等待您输入搜寻的字符串了。此时，输入 date 后， man page 就会开始搜寻跟 date 有关的字符串， 并且移动到该区域呢！很方便吧！最后，如果要离开 man page 时，直接按下『 q 』就能够离开了。 我们将一些在 man page 常用的按键给他整理整理：

按键	进行工作
空格键	向下翻一页
[Page Down]	向下翻一页
[Page Up]	向上翻一页
[Home]	去到第一页
[End]	去到最后一页
/string	向『下』搜寻 string 这个字符串，如果要搜寻 vbird 的话，就输入 /vbird
?string	向『上』搜寻 string 这个字符串
n, N	利用 / 或 ? 来搜寻字符串时，可以用 n 来继续下一个搜寻 (不论是 / 或 ?) ，可以利用 N 来进行『反向』搜寻。举例来说，我以 /vbird 搜寻 vbird 字符串， 那么可以 n 继续往下查询，用 N 往上查询。若以 ?vbird 向上查询 vbird 字符串， 那我可以用 n 继续『向上』查询，用 N 反向查询。
q	结束这次的 man page

要注意喔！上面的按键是在 man page 的画面当中才能使用的！ 比较有趣的是那个搜寻啦！我们可以往下或者是往上搜寻某个字符串，例如要在 man page 内搜寻 vbird 这个字符串， 可以输入 /vbird 或者是 ?vbird ，只不过一个是往下而一个是往上来搜寻的。而要 重复搜寻 某个字符串时，可以使用 n 或者是 N 来动作即可呢！ 很方便吧！^_^

既然有 man page ，自然就是因为有一些文件数据，所以才能够以 man page 来读出来啰！ 那么这些 man page 的数据 放在哪里呢？不同的 distribution 通常可能有点差异性，不过，通常是放在 /usr/share/man 这个目录里头，然而，我们可以透过修改他的 man page 搜寻路径来改善这个目录的问题！修改 /etc/man.config (有的版本为 man.conf 或 manpath.conf) 即可啰！至于更多的关于 man 的讯息您可以使用『 man man 』来查询呦！关于更详细的设定，我们会在 Shell 的章节当中继续的说明喔！

man 还有一些有趣的使用方式呢！举例来说，如果您还想要知道更多跟 man 有较相关的讯息， 可以下达：

```
[root@linux ~]# man -f man
man                  (1)  - format and display the on-line manual pages
man                  (7)  - macros to format man pages
man.conf [man]       (5)  - configuration data for man
```

看到了吗？使用 -f 的参数，可以取得更多的 man 的相关信息，而上头这个表格当中，也有提示了 (数字) 的内容， 举例来说，第二行的『 man (7) 』表示有个 man (7) 的说明文件存在喔！但是却有个 man (1) 存在啊！ 那当我们下达『 man man 』的时候，到底是找到哪一个说明档呢？嘿嘿！混乱了吧？！ 其实，您可以指定不同的文件的，举例来说，上表当中的两个 man 您可以这样将他的文件叫出来：

```
[root@linux ~]# man 1 man  <==这里是用 man(1) 的文件数据
[root@linux ~]# man 7 man  <==这里是用 man(7) 的文件数据
```

你可以自行将上面两个指令输入一次看看，就知道，两个指令输出的结果是不同的。 那个 1, 7 就是分别取出在 man page 里面关于 1 与 7 相关数据的文件档案啰！ 好了，那么万一我真的忘记了下达数字，只有输入『 man man 』时，那么取出的数据到底是 1 还是 7 啊？ 这个就跟搜寻的顺序有关了。搜寻的顺序是记录在 /etc/man.conf 这个设定档当中， 先搜寻到的那个说明档，就会先被显示出来！ 一般来说，通常会先找到数字较小的那个啦！因为排序的关系啊！所以， man man 会跟 man 1 man 结果相同！ 这样说，可以明白了吗？！

除此之外，我们还可以利用『关键词』找到更多的说明文件数据喔！例如：

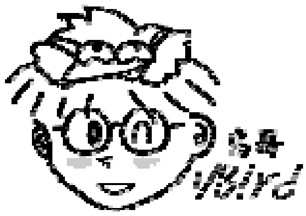
```
[root@linux ~]# man -k man
. [builtins]         (1)  - bash built-in commands, see bash(1)
alias [builtins]     (1)  - bash built-in commands, see bash(1)
........(中间省略)....
xsm                  (1x)  - X Session Manager
zshall               (1)  - the Z shell meta-man page
zshbuiltins          (1)  - zsh built-in commands
zshzle               (1)  - zsh command line editor
```

看到了吧！很多对吧！因为这个是利用关键词将说明文件里面只要含有 man 那个字眼的(不见得是完整字符串) 就将他取出来！很方便吧！ ^_^

事实上，还有两个指令与 man page 有关呢！而这两个指令是 man 的简略写法说～ 就是这两个：

```
[root@linux ~]# whatis  [指令或者是数据]   <==相当于 man -f [指令或者是数据]
[root@linux ~]# apropos [指令或者是数据]   <==相当于 man -k [指令或者是数据]
```

Tips:
一般来说，鸟哥是真的不会去背指令的，只会去记住几个常见的指令而已。那么鸟哥是怎么找到所需要的指令呢？ 举例来说，打印的相关指令，鸟哥其实仅记得 lp (line print)而已。那我就由 man lp 开始，去找相关的说明， 然后，再以 lp[tab][tab] 找到任何以 lp 为开头的指令，找到我认为可能有点相关的指令后， 再以 man 去查询指令的用法！呵呵！所以，如果是实际在管理 Linux ，那么真的只要记得几个很重要的指令即可，其它需要的，嘿嘿！努力的找男人(man)吧！



info page

在所有的 Unix Like 系统当中，都可以利用 man 来查询指令或者是相关档案的用法； 但是，在 Linux 里面则又额外提供了一种在线求助的方法，那就是利用 info 这个好用的家伙啦！ 基本上， info 与 man 其实差不多，而且，文件数据必须要以 info 写成的，才会比较完整。 而这个支持 info 指令的文件是放置在 /usr/share/info/ 这个目录当中的。 举例来说，info 的说明文件有写成 info 格式，所以，你使用 info info 可以得到：

```
[root@linux ~]# info info
File: info.info,  Node: Top,  Next: Getting Started,  Up: (dir)

Info: An Introduction
*********************

The GNU Project distributes most of its on-line manuals in the "Info
format", which you read using an "Info reader".  You are probably using
an Info reader to read this now.

   There are two primary Info readers: `info', a stand-alone program
designed just to read Info files, and the `info' package in GNU Emacs,
a general-purpose editor.  At present, only the Emacs reader supports
using a mouse.

   If you are new to the Info reader and want to learn how to use it,
type the command `h' now.  It brings you to a programmed instruction
sequence.

   To read about expert-level Info commands, type `n' twice.  This
brings you to `Info for Experts', skipping over the `Getting Started'
chapter.

* Menu:

* Getting Started::             Getting started using an Info reader.
```

```
* Expert Info::                Info commands for experts.
* Creating an Info File::      How to make your own Info file.
* Index::                      An index of topics, commands, and variables.

--zz-Info: (info.info.gz)Top, 29 lines --All----------------------------------
Welcome to Info version 4.8. Type ? for help, m for menu item.
```

仔细的看到上面这个表,您可以发现最后一行显示出目前的 info 这个程序的版本信息, 你可以按下 m 这个按键,就可以有更多的指令说明。 而第一行则显示目前这个 info page 的檔名, 注意到我将他显示成为特殊字体的那几个部分,第一行的 Node 显示,这个画面是『在第几层?』的意思, 因为 info page 将所有有关的资料都进行了连结,因此,他可以利用分层的架构来说明每个文件数据呢! 而且还有下一层数据,因此,您会看到第一行还有 Next 这个字眼。这表示, 您只要输入『n』这个按键后,就可以跑到下一层, 也就是 Getting Started 那个章节去了!呵呵!很方便吧!

再来,你也会看到有『Menu』那个咚咚吧!底下共分为四小节,分别是 Getting Started 等等的, 我们可以将光标移动到该文字或者 * 上面,按下 Enter , 就可以前往该小节了!而,利用 [Tab] 按键,就可以快速的将光标在上表的画面中的 node 间移动,真的是很方便啦!不过,什么是 node 呢?就是各个入口点称为 node 。举例来说,上个表格当中,按下 n 或者是将游标游动到 Next 这个字上,按下 Enter ,就可以前往下个说明了。 这就是 node 啊!

不过,就如同前面说的, info 需要文件有支持才行,如果我们以没有支持的 man 来看的话, info man 的结果与 man man 的结果就一样了~没有不同啊!

至于 info page 当中可以使用的按键,可以整理成这样:

按键	进行工作
空格键	向下翻一页
[Page Down]	向下翻一页
[Page Up]	向上翻一页
[tab]	在 node 之间移动,有 node 的地方,通常会以 * 显示。
[Enter]	当光标在 node 上面时,按下 Enter 可以进入该 node 。
b	移动光标到该 info 画面当中的第一个 node 处
e	移动光标到该 info 画面当中的最后一个 node 处
n	前往下一个 info page 处
p	前往上一个 info page 处
u	向上移动一层
s(/)	在 info page 当中进行搜寻
h	显示求助选单
?	指令一览表
q	结束这次的 info page

info page 也是很不错用啦!有兴趣的话,可以多多去查询查询哩! ^_^

其它有用的文件(documents)

刚刚前面说，一般而言，指令或者软件制作者，都会将自己的指令或者是软件的说明制作成『在线说明文件』！ 但是，毕竟不是每个咚咚都需要做成在线说明文件的，还有相当多的说明需要额外的文件！此时，这个所谓的 How-To (如何做的意思) 就很重要啦！还有，某些软件不只告诉你『如何做』， 还会有一些相关的原理会说明，那么这些说明文件要摆在哪里呢？哈哈！就是摆在这个目录 /usr/share/doc 啦！所以说，其实，您只要到这个目录底下，就会发现好多好多的说明文件档啦！还不需要到网络上面找数据呢！厉害吧！ ^_^

举例来说，您想要知道这一版的 Fedora 相关的各项信息，可以直接到：

- /usr/share/doc/fedora-release-4

这个目录来查阅一下即可了解！如果想要知道 bash 是什么，则可以到 /usr/share/doc/bash-3.0 这个目录中！ 很多东西呦！而且， /usr/share/doc 这个目录下的数据主要是以套件 (packages) 为主的， 例如 GCC 这个套件的相关信息在 /usr/share/doc/gcc-xxx(那个 xxx 表示版本的意思！)。 未来可得多多查阅这个目录喔！ ^_^

记住喔！在文字接口下，有任何你不知道的玩意儿，但是你想要了解他，请赶快使用 man 或者是 info 来查询这个玩意儿！此外，如果你想要架设一些其它的服务时，请赶快到 /usr/share/doc 底下查一查有没有该服务的说明档喔！另外，再次的强调，因为 Linux 毕竟是外国人发明的，所以中文文件确实是比较少的！但是不要害怕，拿本英文字典在身边吧！ 随时查阅！不要害怕英文喔！

正确的关机方法(shutdown, reboot, init, halt)

OK！大概知道开机的方法，也知道基本的指令操作，而且还已经知道在线查询了，好累呦！ 想去休息呢！那么如何关机呢？我想，很多朋友在 DOS 的年代已经有在玩计算机了！在当时我们关掉 DOS 的系统时，常常是直接关掉电源开关，而 Windows 在你不爽的时候，按着电源开关四秒也可以关机！但是在 Linux 则相当的不建议这么做！

Why？在 Windows(非 NT 主机系统) 系统中，由于是单人假多任务的情况，所以即使你的计算机关机， 对于别人应该不会有影响才对！不过呢，在 Linux 底下，由于每个程序（或者说是服务） 都是在在背景下执行的，因此，在你看不到的屏幕背后其实可能有相当多人同时在你的主机上面工作， 例如浏览网页啦、传送信件啦以 FTP 传送档案啦等等的，如果你直接按下电源开关来关机时， 则其它人的数据可能就此中断！那可就伤脑筋了！此外，最大的问题是， 若不正常关机，则可能造成档案系统的毁损 （因为来不及将数据回写到档案中，所以有些服务的档案会有问题！）。正常情况下，要关机时需要注意底下几件事：

- 观察系统的使用状态：如果要看目前有谁在在线，可以下达 who 这个指令，而如果要看网络的联机状态，可以下达 netstat -a 这个指令，而要看背景执行的程序可以执行 ps -aux 这个指令。使用这些指令可以让你稍微了解主机目前的使用状态！当然啰，就可以让你判断是否可以关机了（这些指令在后面 Linux 常用指令中会提及喔！）
- 通知在线使用者关机的时刻： 要关机前总得给在线的使用者一些时间来结束他们的工作，所以，这个时候你可以使用 shutdown 的特别指令来达到此一功能。
- 正确的关机指令使用：例如 shutdown 与 reboot 两个指令！

所以底下我们就来谈一谈关于这个关机的正确指令用法啰！

- 将数据同步写入硬盘中的指令： sync
- 惯用的关机指令： shutdown
- 重新开机，关机： reboot, halt, poweroff

数据同步写入磁盘： sync

在 Linux 系统中，为了加快数据的读取速度，所以，预设的情况中， 某些数据将不会直接被写入硬盘，而是先暂存在内存当中，如此一来， 如果一个数据被你重复的改写，那么由于他尚未被写入硬盘中，因此可以直接由内存当中读取出来， 在速度上一定是快上相当多的！

不过，如此一来也造成些许的困扰，那就是，万一当你重新开机，或者是关机，或者是不正常的断电的情况下， 由于数据尚未被写入硬盘当中，哇！所以就会造成数据的更新不正常啦！那要怎么办呢？这个时候就需要 sync 这个指令来进行数据的写入动作啦！直接在文字接口下输入 sync ， 那么在内存中尚未被更新的数据，就会被写入硬盘中！所以，这个指令在系统关机或重新开机之前， 很重要喔！最好多执行几次！(注：这个指令也只有 root 可以执行喔！)

虽然目前的 shutdown/reboot/halt 等等指令均已经在关机前进行了 sync 这个工具的呼叫， 不过，多做几次总是比较放心点～呵呵～

```
[root@linux ~]# sync
```

惯用的关机指令： shutdown

好了，由于关机有种种的限制因子在，所以只有 root 有权力关机而已喔！ 嗯！那么就来关机试试看吧！我们较常使用的是 shutdown 这个指令，而这个指令会通知系统内的各个程序(processes)，并且将通知系统中的 run-level 内的一些服务来关闭(run-level 会在后面告知喔)。shutdown 可以达成：

- 可以自由选择关机模式：是要关机、重新开机或进入单人操作模式均可；
- 可以设定关机时间: 可以设定成现在立刻关机, 也可以设定某一个特定的时间才关机。
- 可以自订关机讯息：在关机之前，可以将自己设定的讯息传送给在线 user 。
- 可以仅发出警告讯息：有时有可能你要进行一些测试，而不想让其它的使用者干扰， 或者是明白的告诉使用者某段时间要注意一下！这个时候可以使用 shutdown 来吓一吓使用者，但却不是真的要关机啦！
- 可以选择是否要 fsck 检查档案系统 。

那么 shutdown 的语法是如何呢？聪明的读者大概已经开始找『男人』了！没错，随时随地的 man 一下，是很不错的举动！好了，简单的语法规则为：

```
[root@linux ~]# /sbin/shutdown [-t 秒] [-arkhncfF] [时间] [警告讯息]
实例：
[root@linux ~]# /sbin/shutdown -h 10 'I will shutdown after 10 mins'
告诉大家，这部机器会在十分钟后关机！并且会显示在目前登入者的屏幕前方！
```

```
至于参数有哪些呢？以下介绍几个吧！

-t sec  ： -t 后面加秒数，亦即『过几秒后关机』的意思
-k      ： 不要真的关机，只是发送警告讯息出去！
-r      ： 在将系统的服务停掉之后就重新开机
-h      ： 将系统的服务停掉后，立即关机。
-n      ： 不经过 init 程序，直接以 shutdown 的功能来关机
-f      ： 关机并开机之后，强制略过 fsck 的磁盘检查
-F      ： 系统重新开机之后，强制进行 fsck 的磁盘检查
-c      ： 取消已经在进行的 shutdown 指令内容。
```

此外，需要注意的是，时间参数请务必加入，否则会自动跳到 run-level 1 （就是单人维护的登入情况），这样就伤脑筋了！底下提供几个例子吧！

```
[root@linux ~]# shutdown -h now
立刻关机，其中 now 相当于时间为 0 的状态
[root@linux ~]# shutdown -h 20:25
系统在今天的 20:25 分会关机
[root@linux ~]# shutdown -h +10
系统再过十分钟后自动关机
[root@linux ~]# shutdown -r now
系统立刻重新开机
[root@linux ~]# shutdown -r +30 'The system will reboot'
再过三十分钟系统会重新开机，并显示后面的讯息。
[root@linux ~]# shutdown -k now 'This system will reboot'
仅发出警告信件的参数！系统并不会关机啦！吓唬人！
```

重新开机，关机： reboot, halt, poweroff

这三个指令差不多，用途上有些不同而已！那个 reboot 其实与 shutdown -r now 几乎相同！不过，建议在关机之前还是将数据回填的指令下达一次再说：

```
[root@linux ~]# sync; sync; sync; reboot
```

就可以啦！通常我如果忘记 shutdown 的指令，或者是怕麻烦，都是使用上面说的这一个指令来重新开机，并且在听到『逼』的一声时，立刻将 Linux 主机的总电源关闭！如此亦可达到关机的目的呀！ 此外， halt 与 poweroff 也具有相同的功能喔！不要担心，使用 man 去查询一下下吧！ ^_^

开机过程的问题排解

事实上， Linux 主机是很稳定的，除非是硬件问题与系统管理员不小心的动作，否则， 很难会造成一些无法挽回的错误的。但是，毕竟我们目前使用的可能是练习机，会常常开开关关的， 所以确实可能会有一些小问题发生。好了，我们先来简单的谈一谈，如果无法顺利开机时， 您应该如何解决。要注意的是，底下说到的内容很多都还没有开始介绍， 因此，看不懂也不要太紧张～在本书全部都读完且看第二遍时，您自然就会有感觉了！ ^_^

扇区错乱的问题

在开机的过程中最容易遇到的问题就是硬盘可能有坏轨或扇区错乱（数据损毁）的情况， 这种情况虽然不容易发生在稳定的 Linux 系统下，不过由于不当的开关机 还是可能会造成的，原因可能有：

- 最可能发生的原因是因为断电或不正常关机所导致的硬盘磁道错乱， 鸟哥的主机就曾经发生过多次因为跳电，家里的主机又没有安装不断电系统， 结果就导致硬盘磁道错乱了！
- 硬盘使用率过高也是一个可能的原因，例如你开放了一个 FTP 服务，里面有些数据很有用， 所以一堆人抢着下载，如果你又不是使用较稳定的 SCSI 接口硬盘，仅使用 IDE 接口的硬盘， 虽然机率真的不高，但还是有可能造成磁道错乱的（这个问题其实在 Windows 系统下更容易发生哩！）。

解决的方法其实很简单，也可能很困难，如果您的根目录『/』并没有损毁，那就很容易解决， 如果根目录已经损毁了，那就比较麻烦！

- 如果根目录没有损毁：假设你发生错误的磁盘区块是在 /dev/hda7 这一块，那么在开机的时候，屏幕应该会告诉你： press root pass word or ctrl+D : 这时候请输入 root 的密码登入系统：
    - 在光标处输入 root 密码登入系统，进行单人单机的维护工作；
    - 输入 fsck /dev/hda7 （fsck 为指令， /dev/hda7 为错误的磁盘区块，请依你的情况下达参数）， 这时屏幕会显示开始修理硬盘的讯息，如果有发现任何的错误时，屏幕会显示： clear [Y/N]？ 的询问讯息，就直接输入 Y 吧！
    - 修理完成之后，以 reboot 重新开机啰！

- 如果根目录损毁了： 一般初学者喜欢将自己的硬盘只划分为一个大区块，亦即只有根目录， 那硬盘区块错乱一定是根目录的问题啰！这时你可以将硬盘拔掉，接到另一台 Linux 系统的计算机上，并且不要挂载（mount）该硬盘，然后以 root 的身份执行 fsck /dev/hdb1 （/dev/hdb1 指的是你的硬盘装置名称，你要依你的实际状况来设定），这样就 OK 啰！

  另外，也可以使用近年来很热门的 Live CD ，也就是利用光盘开机就能够进入 Linux 操作系统的特性， 您可以前往：『http://knoppix.tnc.edu.tw/』 这个网站来下载，并且烧录成为 CD ，这个时候用光盘开机，再加以 mount 原本的 / ， 以 fsck /dev/hda1 就能够就回来了！
- 如果硬盘整个坏掉： 如果硬盘实在坏的离谱时，那就换一颗吧，如果不愿意换硬盘，那就重灌 Linux 吧，并且在重灌的时候，于 Format 项目中，选取『error check』项目，只是如此一来，format 会很慢，并且，何时会再坏掉也不确定！最好还是换一颗吧！

预防的方法：

- 妥善保养硬盘：例如： 主机通电之后不要搬动，避免移动或震动硬盘；尽量降低硬盘的温度，可以加装风扇来冷却硬盘； 或者可以换装 SCSI 硬盘。

- 划分不同的磁盘区块：如果诸位看官还记得的话，应该知道 Fedora 安装的方法可以分为四五种，有 upgrad, Server, Workstation 等等的，这些咚咚不一样的地方就在于磁盘划分的不同啦！通常我们会建议划分下列的磁盘区块：
    - /
    - /boot
    - /usr
    - /home
    - /var

  这样划分有些好处，例如 /var 是系统预设的一些数据暂存或者是 cache 数据的储存目录， 像 e-mail 就含在这里面。如果还有使用 proxy 时，因为常常存取，所以有可能会造成磁盘损坏， 而当这部份的磁盘损坏时，由于其它的地方是没问题的，因此资料得以保存，而且在处理时也比较容易！

忘记 root 密码：

常常有些朋友在设定好了 Linux 之后，结果 root 密码给他忘记去！要重新安装吗？！ 不需要的，你只要以单人维护模式登入即可更改你的 root 密码喔！不过，目前的开机选单主要有 LILO 与 Grub 两种，这两种模式并不相同，有必要来说明一下：

- LILO

您只要在出现 Lilo 选单的时候，输入：（请注意，如果是 Red Hat 7.0 以后的版本，会出现图形接口的 lilo ，这个时候请按下 [Ctrl] + x 即可进入纯文字接口的 lilo 啰！）

```
boot: linux -s
```

以进入单人单机维护模式(即为 run-level 1)，然后再输入 passwd 这个指令，就可以直接更改 root 的密码啰！同时，如果图形接口无法登入的时候，也可以使用此一方法来进入单人单机的维护工作，然后再去修改 /etc/inittab 改变一下登入的预设模式，如此一来就可以在下次开机的时候以文字模式登入啰！同时请注意！如果您在设定启动的名称的时候，已经改变了启动的名称，例如我都喜欢在名称之后加上核心码， 例如：Red-Hat-2.4.7，这个时候我就必须在 boot: 底下输入：

```
boot: Red-Hat-2.4.7linux -s
boot: Red-Hat-2.4.7linux single
```

另外，您可能会遇到 Lilo 的开机问题，这个时候建议您先参考一下底下这一篇讨论，未来还会再次提到 lilo 的设定问题！

- http://phorum.vbird.org/viewtopic.php?t=150

- Grub

grub 做为开机管理程序，要进入单人维护模式就比较麻烦一些！ 在开机的过程当中，会有读秒的时刻，此时请按下任意按键，就会进入选单画面。 这个时候只要选择相对的核心档案，并且按下『e』，就可以进入编辑画面了。此时，你看到的画面有点像：

```
root    (hd0,0)
```

```
kernel /boot/vmlinuz-2.4.19 root=/dev/hda1
```

此时，请将光标移动到 kernel 那一行，再按一次『 e 』进入 kernel 该行的编辑画面中， 然后在出现的画面当中，最后方输入 single ：

```
root    (hd0,0)
kernel /boot/vmlinuz-2.4.19 root=/dev/hda1 single
```

再按下『 Enter 』确定之后，按下 b 就可以 boot 看看啦！

关于 LILO 与 grub 我们会在后面继续加以介绍，这里先有概念即可！不过，并非所有版本的 Linux 皆可使用此一方法！例如 OpenLinux 就无法进入单人维护模式，因为他的单人维护模式 『仍然需要输入 root 的密码！』哇！真是可怕呐！这个时候怎么办？呵呵！别担心，现在你还不会， 看完了后面几个章节之后，您就一定会啦！ ^_^

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 请问如果我以文字模式登入 Linux 主机时，我有几个终端机接口可以使用？如何切换各个不同的终端机接口？

  共有六个， tty1 ~ tty6 ，切换的方式为 Crtl + Alt + [F1]~[F6]，其中， [F7] 为图形接口的使用。

- 在 Linux 系统中， /VBird 与 /vbird 是否为相同的档案？

  两者为不同的档案，因为 Linux 系统中，大小写字母代表意义不一样！

- 我想要知道 date 如何使用，应该如何查询？

  最简单的方式就是使用 man date 或 info date 来查看，如果该套件有完整说明的话，那么应该也可以在 /usr/share/doc 里面找到说明档！

- 我想要在今天的 1:30 让系统自己关机，要怎么做？

  shutdown -h 1:30

- 如果我 Linux 的 X Window 突然发生问题而挂掉，但 Linux 本身还是好好的，那么我可以按下哪三个按键来让 X window 重新启动？

  [crtl]+[alt]+[backspace]

- man page 的设定档在哪里？

  Fedora 的设定档在 /etc/man.conf 有的 distribution 会定义为 /etc/man.config 或 /etc/manpath.conf

- 我想要知道 2005 年 5 月 2 日是星期几？该怎么做？

  最简单的方式直接使用 cal 5 2005 即可找出 2003 年 5 月份的月历。

- 使用 man date 然后找出显示目前的日期与时间的参数，成为类似：2002/10/16-20:03

  date +%Y/%m/%d-%H:%M

- 若以 X-Window 为预设的登入方式，那请问如何进入 Virtual console 呢？

  可以按下 [Ctrl] + [Alt] + [F1] ~ [F6] 进入 Virtual console (共六个)； 而按下 [Ctrl] + [Alt] + [F8] 或 [F7] 可回到 X-Window 的 desktop 中！

- 简单说明在 bash shell 的环境下， [tab] 按键的用途？

  [Tab] 按键可做为命令补齐或档案补齐的功能，与所接的指令位置有关。 接在一串指令的第一个单字后面，则为命令补齐，否则则为档案补齐！

- 如何强制中断一个程序的进行？(利用按键，非利用 kill 指令)

  可以利用 [Ctrl] + c 来中断！

- Linux 提供相当多的在线查询，称为 man page，请问，我如何知道系统上有多少关于 passwd 的说明？又，可以使用其它的程序来取代 man 的这个功能吗？

  可以利用 man -f passwd 来查询，另外，如果有提供 info 的文件数据时 (在 /usr/share/info/ 目录中) ，则能够利用 info passwd 来查询之！

- man -k passwd 与 man -K passwd 有什么差异(大小写的 K)？

  小写的 -k 为查询关键词，至于 -K 则是整个系统的 man page 查询～ 每个被检查到有关键词的 man page file 都会被询问是否要显示， 您可以输入『ynq』，来表示：y:要显示到屏幕上；n:不显示；q:结束 man 的查询。

- 在 man 的时候， man page 显示的内容中，指令(或档案)后面会接一组数字，这个数字若为 1, 5, 8 ，表示该查询的指令(或档案)意义为何？

  代表意义为 1) 一般使用者可以使用的指令或可执行档案 5)一些设定档的档案内容格式 8)系统管理员能够使用的管理指令。

- man page 显示的内容的档案是放置在哪些目录中？

  放置在 /usr/share/man/ 与 /usr/local/man 等预设目录中。

- 请问这一串指令『 foo1 -foo2 foo3 foo4 』中，各代表什么意义？

foo1 一定是指令， -foo2 则是 foo1 这个指令的选择项目参数， foo3 与 foo4 则不一定， 可能是 foo1 的参数设定值，也可能是额外加入的 parameters。

- 当我输入 man date 时，在我的终端机却出现一些乱码，请问可能的原因为何？如何修正？

  如果没有其它错误的发生，那么发生乱码可能是因为语系的问题所致。 可以利用 LANG=en 或者是 LANG=en_US 等设定来修订这个问题。

- 我输入这个指令『ls -al /vbird』,系统回复我这个结果:『ls: /vbird: No such file or directory』请问发生了什么事？』

  不要紧张，很简单的英文，因为系统根本没有 /vbird 这个档案的存在啊！ ^_^

- 你目前的 Linux 底下，预设共有多少可以被你执行的指令？

  最简单的做法，直接输入两次 [tab] 按键即可知道有多少指令可以被执行。

- 我想知道目前系统有多少指令是以 bz 为开头的，可以怎么作？

  直接输入 bz[tab][tab] 就可以知道了！

- 承上题，在出现的许多指令中，请问 bzip2 是干嘛用的？

  在使用 man bzip2 之后，可以发现到，其实 bzip2 是用来作为压缩与解压缩档案用的！

- Linux 提供一些在线文献数据，这些数据通常放在那个目录当中

  通常放在 /usr/share/doc 当中！

- 在终端机里面登入后，看到的提示字符 $ 与 # 有何不同？平时操作应该使用哪一个？

  # 代表以 root 的身份登入系统，而 $ 则代表一般身份使用者。依据提示字符的不同， 我们可以约略判断登入者身份。一般来说，建议日常操作使用一般身份使用者登入，亦即是 $ ！

Linux 的档案属性与目录配置

最近更新日期：2006/08/05

Linux 最优秀的地方之一，就在于他的多人多任务的环境。 而为了让各个使用者具有较安全的管理机制，因此档案的权限管理就变的很重要了。 Linux 一般将档案可存取的方式分为三个类别，分别是 owner/group/other， 且各有 read/write/excute 等权限。若管理得当，将会让您的 Linux 主机变的较为安全。另外，您如果首次接触 Linux 的话，那么， 在 Linux 底下这么多的目录/档案，到底代表什么意义呢？ 底下我们就来一一介绍呢！

使用者与群组

经过前面一章的洗礼之后，您应该可以在 Linux 的指令列模式底下输入指令了吧？！ 呵呵！接下来，当然是要让您好好的浏览一下 Linux 系统里面有哪些重要的档案啰。 不过，每个档案都有相当多的属性，其中最重要的可能就是档案的拥有者的概念了。 所以，在开始档案相关信息的介绍前，鸟哥先就简单的使用者及群组的概念作个说明吧～ 好让您快点进入状况的哩！ ^_^

- 档案拥有者

初次接触 Linux 的朋友大概会觉得很怪异，怎么『Linux 有这么多使用者，还分什么群组，有什么用？』。这个『使用者与群组』的功能可是相当健全而好用的一个安全防护呢！怎么说呢？ 由于 Linux 是个多人多任务的系统 (已经提过若干次啰！),因此可能常常会有多人同时使用这部主机来进行工作的情况发生，为了考虑每个人的隐私权以及每个人的喜好的工作环境， 因此，这个『档案拥有者』的角色就显的相当的重要了！

例如当你将你的 e-mail 情书转存成档案之后，放在您自己的家目录，您总不希望被其它人看见自己的情书吧？ 这个时候，你就把该档案设定成『只有档案拥有者，就是我，才能看与修改这个档案的内容』， 那么即使其它人知道你有这个相当『有趣』的档案，不过由于您有设定适当的权限， 所以其它人自然也就无法知道该档案的内容啰！

- 群组概念

那么群组呢？为何要设定档案还有所属的群组？其实， 群组最简单的功能之一，就是当您在团队开发资源的时候最有用啦！ 举个例子来说好了，假如在我的主机上面有两个团体，这第一个团体名称为 testgroup 而他的成员是 test1, test2, test3 三个，第二个团体名称为 treatgoup 他的团员为 treat1, treat2, treat3，这两个团体之间是互相有竞争性质的， 但是却又要缴交同一份报告，然而每组团员又需要同时可以修改自己的团体内任何人所建立的档案， 且不能让非自己团体的其它人看到自己的档案内容！这个时候怎么办？

呵呵！在 Linux 底下可就很简单啦！我可以经由简易的档案权限设定，就能限制非自己团队(亦即是群组啰)的其它人不能够阅览内容啰！而且亦可以让自己的团队成员可以修改我所建立的档案！ 同时，如果我自己还有私人隐密的文件，仍然可以设定成让自己的团队成员也看不到我的档案数据。 很方便吧！

另外，如果 teacher 这个账号是 testgroup 与 treatgroup 这两个群组的老师， 他想要同时观察两者的进度，因此需要两边的群组都能够进去观看，这个时候，您可以设定 teacher 这个账号，『同时支持 testgroup 与 treatgroup 这两个群组！』，也就是说， 每个人都可以有多个群组的支持呢！

这样说或许你还不容易理解这个使用者与群组的关系吧？没关系，我们可以使用目前『家庭』的观念来进行解说喔！ 假设有一家人，家里只有三兄弟，分别是王大毛、王二毛与王三毛三个人， 而这个家庭是登记在王大毛的名下的！所以，『王大毛家有三个人，分别是王大毛、王二毛与王三毛』， 而且这三个人都有自己的房间，并且共同拥有一个客厅喔！

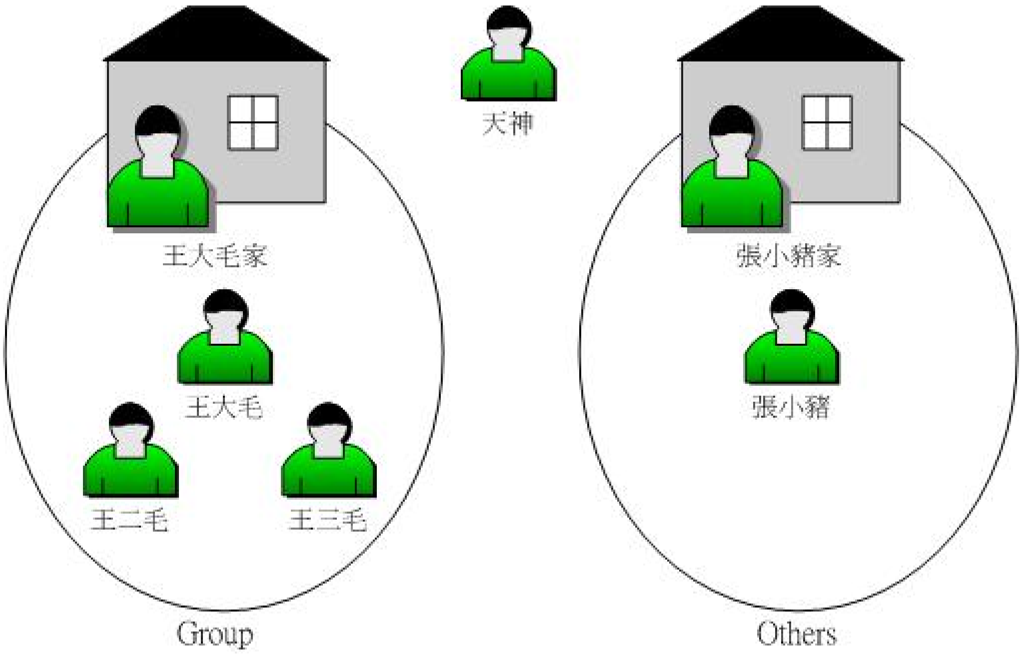
- 由于王家三个人各自拥有自己的房间，所以，王二毛虽然可以进入王三毛的房间， 但是二毛不能翻三毛的抽屉喔！那样会被三毛 K 的！因为抽屉里面可能有三毛自己私人的东西，例如情书啦，日记啦等等的， 这是『私人的空间』，所以当然不能让二毛拿啰！
- 由于共同拥有客厅，所以王家三兄弟可以在客厅打开电视机啦、翻阅报纸啦、坐在沙发上面发呆啦等等的！ 反正，只要是在客厅的玩意儿，三兄弟都可以使用喔！因为大家都是一家人嘛！

这样说来应该有点晓得了喔！那个『王大毛家』就是所谓的『群组』啰， 至于三兄弟就是分别为三个『使用者』，而这三个使用者是在同一个群组里面的喔！ 而三个使用者虽然在同一群组内，但是我们可以设定『权限』， 好让某些使用者个人的信息不被群组的所有人查询，以保有个人『私人的空间』啦！ 而设定群组共享，则可让大家共同分享喔！

- 其它人的概念

好了，那么今天又有个人，叫做张小猪，他是张小猪家的人，与王家没有关系啦！ 这个时候，除非王家认识张小猪，然后开门让张小猪进来王家，否则张小猪永远没有办法进入王家， 更不要说进到王三毛的房间啦！不过，如果张小猪透过关系认识了三毛，并且跟王三毛成为好朋友， 那么张小猪就可以透过三毛进入王家啦！呵呵！没错！那个张小猪就是所谓的『其它人， Others 』啰！

因此，我们就可以知道啦，在 Linux 里面，任何一个档案都具有『User, Group 及 Others』三个权限！ 我们可以将上面的说明以底下的图示来解释：



图一、每个档案的拥有者、群组与其它人的示意图

此时，以王三毛为例，王三毛这个『档案』的拥有者为王三毛，他属于王大毛这个群组， 而张小猪相对于王三毛，则只是一个『其它人(others)』而已。

不过，这里有个特殊的人物要来介绍的，那就是『万能的天神』！这个天神具有无限的神力， 所以他可以到达任何他想要去的地方，呵呵！那个人在 Linux 系统中的身份代号是『 root 』啦！所以要小心喔！那个 root 可是『万能的天神』喔！

无论如何，『使用者身份』，与该使用者所支持的『群组』概念，在 Linux 的世界里面是相当的重要的，他可以帮助您让您的多任务 Linux 环境变的更容易管理！更详细的 『身份与群组』 设定，我们将在 账号管理 再进行解说。底下我们将针对档案系统与档案权限来进行说明。

- Linux 使用者身份与群组记录的档案

在我们 Linux 系统当中，预设的情况下，所有的系统上的账号与一般身份使用者，还有那个 root 的相关信息，都是记录在 /etc/passwd 这个档案内的。至于密码则是记录在 /etc/shadow 这个档案下。 此外，Linux 所有的群组名称都纪录在 /etc/group 内！这三个档案可以说是 Linux 系统里面账号、密码、群组信息的集中地啰！不要随便删除这三个档案啊！ ^_^

至于更多的与账号群组有关的设定，还有这三个档案的格式，不要急，我们在第四篇讲到账号时， 会再跟大家详细的介绍的！这里先有概念即可。

Linux 档案权限概念

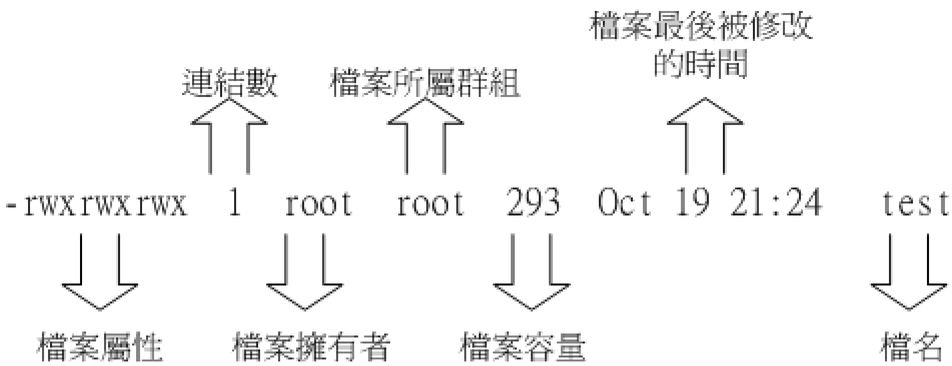
大致了解了 Linux 的使用者与群组之后，接着下来，我们要来谈一谈， 那么这个档案的权限要如何针对这些所谓的『使用者』与『群组』来设定该档案的权限呢？ 这个部分是相当的重要的，尤其对于初学者来说，因为档案的权限与属性是学习 Linux 的一个相当重要的关卡，如果没有这部份的概念，那么您将老是听不懂别人在讲什么呢！ 尤其是当您在您的屏幕前面出现了『Permission deny』的时候，不要担心，『肯定是权限设定错误』啦！呵呵！好了，闲话不多聊，赶快来瞧一瞧先：

Linux 档案属性

嗯！既然要让你了解 Linux 的档案属性，那么有个重要的也是常用的指令就必须要先跟你说啰！那一个？！就是『 ls 』这一个 list 档案的指令啰！在你以 root 的身份登入 Linux 之后，下达『ls -al 』看看，会看到底下的几个咚咚：

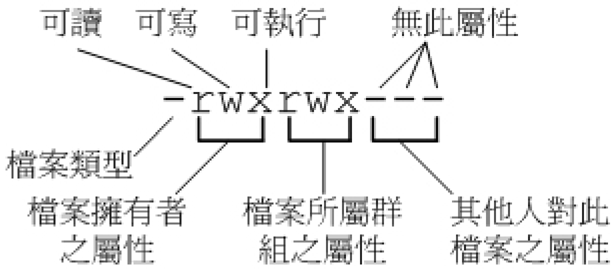
```
[root@linux ~]# ls -al
total 248
drwxr-x---    9   root    root     4096    Jul  11 14:58    .
drwxr-xr-x   24   root    root     4096    Jul   9 17:25    ..
-rw-------    1   root    root     1491    Jun  25 08:53    anaconda-ks.cfg
-rw-------    1   root    root    13823    Jul  10 23:12    .bash_history
-rw-r--r--    1   root    root       24    Dec   4  2004    .bash_logout
-rw-r--r--    1   root    root      191    Dec   4  2004    .bash_profile
-rw-r--r--    1   root    root      395    Jul   4 11:45    .bashrc
-rw-r--r--    1   root    root      100    Dec   4  2004    .cshrc
drwx------    3   root    root     4096    Jun  25 08:35    .ssh
-rw-r--r--    1   root    root    68495    Jun  25 08:53    install.log
-rw-r--r--    1   root    root     5976    Jun  25 08:53    install.log.syslog
[    1    ][ 2 ][  3   ][  4 ][    5    ][      6      ][       7          ]
[  属性   ][连结][拥有者][群组][档案容量][  修改日期   ][      檔名         ]
```

ls 是『list』的意思，与在早期的 DOS 年代的指令 dir 类似功能。 而参数『-al』则表示列出所有的档案（ 包含隐藏档，就是档名前面第一个字符为 . 的那种档案 ）。如上所示，在你第一次以 root 身份登入 Linux 时，如果你输入指令后，应该有上列的几个东西， 先解释一下上面七个字段个别的意思：



图二、档案属性的示意图

1. 第一栏代表这个档案的属性： 这个地方最需要注意了！仔细看的话，你应该可以发现这一栏其实共有十个属性：



图三、档案的十个属性内容

- 第一个属性代表这个档案是『目录、档案或连结文件等等』：
  - 当为[d]则是目录，例如上表的第 11 行；
  - 当为[-]则是档案，例如上表的第 5 行；
  - 若是[l]则表示为连结档(link file)；
  - 若是[b]则表示为装置文件里面的可供储存的接口设备；
  - 若是[c]则表示为装置文件里面的串行端口设备，例如键盘、鼠标。

- 接下来的属性中，三个为一组，且均为『rwx』 的三个参数的组合。其中，[r]代表可读(read)、[w]代表可写(write)、[x]代表可执行(excute)：
  - 第一组为『拥有人的权限』，以第 5 行为例，该档案的拥有人可以读写，但不可执行；
  - 第二组为『同群组的权限』；
  - 第三组为『其它非本群组的权限』。

范例：若有一个档案的属性为『-rwxr-xr--』，简单的可由下面说明之：

[-][rwx][r-x][r--]
 1 234 567 890
 1 为：代表这个文件名为目录或档案（上面为档案）
 234 为：拥有人的权限（上面为可读、可写、可执行）
 567 为：同群组使用者权限（上面为可读可执行）
 890 为：其它使用者权限（上面为仅可读）

上面的属性情况代表一个档案、这个档案的拥有人可读可写可执行、但同群组的人仅可读与执行，非同群组的使用者仅可读的意思！

除此之外，需要特别留意的是 x 这个标号！ 若文件名为一个目录的时候，例如上表中的 .ssh 这个目录：

```
drwx------    3    root     root     4096     Jun 25 08:35     .ssh
```

可以看到这是一个目录，而且只有 root 可以读写与执行。但是若为底下的样式时，请问非 root 的其它人是否可以进入该目录呢？

```
drwxr--r--    3    root     root     4096     Jun 25 08:35     .ssh
```

咦！似乎好像是可以喔！因为有可读[r]存在嘛！『错！』答案是非 root 这个账号的其它使用者均不可进入 .ssh 这个目录，为什么呢？ 因为 x 与 目录 的关系相当的重要， 如果您在该目录底下不能执行任何指令的话，那么自然也就无法进入了，因此， 请特别留意的是，如果您想要开放某个目录让一些人进来的话， 请记得将该目录的 x 属性给开放呦！至于目录的权限相关说明， 我们会在底下继续介绍的。

另外，你也必须要更加的小心的是，在 Windows 底下一个档案是否具有执行的能力是藉由『 附檔名 』来判断的， 例如：.exe, .bat, .com 等等，但是在 Linux 底下， 我们的档案是否能执行，则是藉由是否具有 x 这个属性来决定的！所以，跟档名是没有绝对的关系的！ 这点还请特别留意呢！稍后我们还会针对目录来稍作说明的

2. 第二栏表示为连结占用的节点 (i-node)： 这个跟连结档 (link file) 比较有关系，我们在未来的章节会再加以介绍的。 如果是目录的话，那么就与该目录下还有多少目录有关。

3. 第三栏表示这个档案（或目录）的『拥有人』。

4. 第四栏表示拥有人的群组。
   这里再次解释一下，在 Linux 中，你的 ID（ 如 root 或 test 等账号均是所谓的 ID ）即是你的身份，而且你还可以附属在一个或多个群组之下，例如刚刚我们上面提到的，你有一个团体 （ 即群组 ）代号为 testgroup ，且这个群体里有三个人，其代号分别是 test1, test2, 与 test3，则这三个人为同一群组即 testgroup！那么如果以上图的的档案属性(-rwxrwx---) 来看，如果该档案属于 test1 所有，那么 test2, test3 亦有读、写、执行的权力，因为他们都属于同一个

group 呀！而如果您不是属于 test1, test2, test3 的任何一个人，也不属于 testgroup 这个群组时，那么您将不具备读、写、执行这个档案的权限了！

5. 第五栏为这个档案的容量大小。

6. 第六栏为这个档案的建档日期或者是最近的修改日期， 分别为月份、日期及时间。请特别留意，如果您是以繁体中文来进行安装您的 Linux 时，那么预设的语系可能会被改为中文。而 由于中文无法显示在文字型态的终端机上面， 所以这一栏会成为怪怪的乱码，这个时候，请修改一下 /etc/sysconfig/i18n 这个档案，里面的『 LC_TIME 』修改为：『 LC_TIME=en 』再储存离开，再登入一次，就可以得到英文字形显示的日期了！ 那么如何修改该档案呢？呵呵！以 root 身份用 vi 修改。 另外，也可以使用『 LANG=en ls -al 』之类的语法来显示。

7. 第七栏为这个档案的档名， 如果档名之前多一个『 . 』，则代表这个档案为『隐藏档』， 例如上表第 6 行的『.bashrc_history』档名即是隐藏档，由于我们有下一个参数为 ls -al，所以连隐藏档都列出来，如果你只输入 ls 则档名有加『 . 』的档案就不会被显示出来！

Tips:
对于更详细的 ls 用法，还记得怎么查询吗？对啦！使用 man ls 或 info ls 去看看他的基础用法去！自我进修是很重要的， 因为『师傅带进门，修行在个人！』，自古只有天才学生，没有天才老师呦！加油吧！^_^



这七个字段的意义是很重要的!务必清楚的知道各个字段代表的意义呢!尤其是第一个字段的十个权限，那是整个 Linux 档案权限的重点之一。底下我们来做几个简单的练习，您就会比较清楚啰！ 假设 test1, test2, test3 同属于 testgroup 这个群组：

例题一：如果有下面的两个档案，请说明两个档案的拥有者与其相关的权限为何？

```
-rw-r--r--   1 root     root          238 Jun 18 17:22 test.txt
-rwxr-xr--   1 test1    testgroup    5238 Jun 19 10:25 ping_tsai
```

答：

- 档案『 test.txt 』的拥有人为 root ，群组为 root 。至于权限方面则只有 root 这个账号可以存取此档案，其它人则仅能读此档案；
- 另一个档案『 ping_tsai 』的拥有人为 test1 ，而群组为 testgroup。其中， test1 可以针对此档案具有可读可写可执行的权力，而同群组的 test2, test3 两个人与 test1 同样是 testgroup 的群组账号，则仅可读可执行但不能写 (亦即不能修改)，至于非 testgoup 这一个群组的人则仅可以读，不能写也不能执行！

例题二：如果我的目录为底下的样式，请问 testgroup 这个群组的成员与其它人(others)是否可以进入本目录？

```
drwxr-xr--   1 test1    testgroup    5238 Jun 19 10:25 groups/
```

答：

- 档案拥有者 test1 可以在本目录中进行任何工作；
- 而 testgroup 这个群组的账号，例如 test2, test3 亦可以进入本目录进行工作， 但是不能在本目录下进行写入的动作；
- 至于 other 的权限中虽然有 r ，但是由于没有 x 的权限，因此 others 的使用者，并不能进入此目录！

- Linux 档案属性的重要性：

与 Windows 系统不一样的是，在 Linux 系统（或者说 Unix-Like 系统）当中，每一个档案都多加了很多的属性进来，尤其是群组的概念，这样有什么用途呢？ 基本上，最大的用途是在『安全性』上面的。举个简单的例子，在你的系统中，关于系统服务的档案通常只有 root 才能读写，或者是执行，例如 /etc/shadow 这一个账号管理的档案，由于该档案记录了你的系统中的所有账号的数据，因此是很重要的一个信息文件，当然不能让任何人读取，只有 root 才能够来读取啰！所以该档案的属性就会成为 [-rw-------]啰！

那么，如果你有一个开发团队，在你的团队中，你希望每个人都可以使用某一些目录下的档案， 而非你的团队的其它人则不予以开放呢？以上面的例子来说，testgroup 的团队共有三个人，分别是 test1, test2, test3 ！那么我就可以将 test1 的档案属性订为 [-rwxrwx---] 来提供给 testgroup 的工作团队使用啰！这可是相当重要的。

再举个例子来说，如果你的目录权限没有作好的话，可能造成其它人都可以在你的系统上面乱搞啰！例如本来只有 root 才能做的开关机、ADSL 的拨接程序、新增或删除使用者等等的指令，若被你改成任何人都可以执行的话， 那么如果使用者不小心给你重新开机啦！重新拨接啦！等等的！那么你的系统不就会常常莫名其妙的挂掉啰！ 而且万一你的使用者的密码被其它不明人士取得的话，只要他登入你的系统就可以轻而易举的执行一些 root 的工作！可怕吧！因此，在你修改你的 linux 档案与目录的属性之前， 一定要先搞清楚，什么是可变的，什么是不可变的！千万注意啰！

如何改变档案权限

好了，我们已经知道档案权限对于一个系统的安全重要性了，也知道档案的权限对于使用者与群组的相关性了， 好了，那么如何修改一个档案的权限呢？又！有多少档案的权限我们可以修改呢？ 其实一个档案的权限很多嘛！大致上我们先介绍几个简单的，例如：群组、拥有者、各种身份的权限等等。

- chgrp ：改变档案所属群组
- chown ：改变档案所属人
- chmod ：改变档案的属性、 SUID 、等等的特性

- 改变所属群组, chgrp

改变一个档案的群组真是很简单的，直接以 chgrp 来改变即可，咦！这个指令就是 change group 的缩写嘛！对啦！这样就很好记了吧！ ^_^。不过，请记得， 要改变成为的群组名称必须要在 /etc/group 里面存在的名称才行，否则就会显示错误！

假设您是以 root 的身份登入 FC4 ，那么在您的家目录内有一个 install.log 的档案， 如何将该档案的

群组改变一下呢？假设您已经知道在 /etc/group 里面已经存在一个名为 users 的群组，但是 testing 这个群组名字就不存在 /etc/group 当中了， 此时改变群组成为 users 与 testing 会有什么现象发生呢？

```
[root@linux ~]# chgrp [-R] dirname/filename ...
参数：
-R : 进行递归( recursive )的持续变更，亦即连同次目录下的所有档案、目录
     都更新成为这个群组之意。常常用在变更某一目录的情况。
范例：
[root@linux ~]# chgrp users install.log
[root@linux ~]# ls -l
-rw-r--r--  1 root users 68495 Jun 25 08:53 install.log
[root@linux ~]# chgrp testing install.log
chgrp: invalid group name `testing' <== 发生错误讯息啰～找不到这个群组名～
```

发现了吗？档案的群组被改成 users 了，但是要改成 testing 的时候， 就会发生错误～注意喔！发生错误讯息还是要努力的查一查错误讯息的内容才好！

- 改变档案拥有者，chown

好了，那么如何改变一个档案的拥有者呢？很简单呀！既然改变群组是 change group ，那么改变拥有者就是 change owner 啰！BINGO，对啦！那就是 chown 这个指令的用途，要注意的是，使用者必须是已经存在系统中的，也就是在 /etc/passwd 这个档案中有纪录的使用者名称才行改变。

chown 的用途还满多的，他还可以顺便直接修改群组的名称呢！此外， 如果要连目录下的所有次目录或档案同时更改档案拥有者的话，直接加上 -R 的参数即可！我们来看看语法与范例：

```
[root@linux ~]# chown [-R] 账号名称 档案或目录
[root@linux ~]# chown [-R] 账号名称:群组名称 档案或目录
参数：
-R : 进行递归( recursive )的持续变更，亦即连同次目录下的所有档案、目录
     都更新成为这个群组之意。常常用在变更某一目录的情况。
范例：
[root@linux ~]# chown bin install.log
[root@linux ~]# ls -l
-rw-r--r--  1 bin  users 68495 Jun 25 08:53 install.log
[root@linux ~]# chown root:root install.log
[root@linux ~]# ls -l
-rw-r--r--  1 root root 68495 Jun 25 08:53 install.log
```

嗯！知道如何改变档案的群组与拥有者了，那么什么时候要使用 chown 或 chgrp 呢？！或许您会觉得奇怪吧？！是的，确实有时候需要变更档案的拥有者的， 最常见的例子就是在 copy 档案给你之外的其它人时，我们使用最简单的 cp 来说明好了：

```
[root@linux ~]# cp 来源档案 目的档案
```

假设您今天要将 .bashrc 这个档案拷贝成为 .bashrc_test ，且是要给 bin 这个人，您可以这样做：

```
[root@linux ~]# cp .bashrc .bashrc_test
```

```
[root@linux ~]# ls -al .bashrc*
-rw-r--r--  1 root root 395 Jul  4 11:45 .bashrc
-rw-r--r--  1 root root 395 Jul 13 11:31 .bashrc_test
```

哇！怎么办？ .bashrc_test 还是属于 root 所有，如此一来，即使你将档案拿给 bin 这个使用者了，那他仍然无法修改的(看属性就知道了吧！)， 所以你就必须要将这个档案的拥有者与群组修改一下啰！知道如何修改了吧！？呵呵！

- 改变九个属性，chmod

档案属性的改变使用的是 chmod 这个指令，但是，属性的设定方法有两种， 分别可以使用数字或者是符号来进行属性的变更。我们就来谈一谈：

- 数字类型改变档案权限

Linux 档案的基本属性就有九个，分别是 owner/group/others 组别的 read/write/excute 属性， 先复习一下刚刚上面提到的数据：

-rwxrwxrwx

这九个属性是三个三个一组的！其中，我们可以使用数字来代表各个属性，各属性的对照表如下：

r:4
w:2
x:1

同一组 (owner/group/others) 的三个属性 (r/w/x) 是需要累加的，例如当属性为 [-rwxrwx---] 则是：

owner = rwx = 4+2+1 = 7
group = rwx = 4+2+1 = 7
others= --- = 0+0+0 = 0

所以等一下我们设定属性的变更时，该属性的数字就是 770 啦！变更属性的指令 chmod 的语法是这样的：

```
[root@linux ~]# chmod [-R] xyz 档案或目录
参数：
xyz : 就是刚刚提到的数字类型的权限属性，为 rwx 属性数值的相加。
-R : 进行递归( recursive )的持续变更，亦即连同次目录下的所有档案、目录
     都更新成为这个群组之意。常常用在变更某一目录的情况。
```

举例来说，如果要将 .bashrc 这个档案所有的属性都打开，那么就下达：

```
[root@linux ~]# ls -al .bashrc
-rw-r--r--  1 root root 395 Jul  4 11:45 .bashrc
[root@linux ~]# chmod 777 .bashrc
[root@linux ~]# ls -al .bashrc
-rwxrwxrwx  1 root root 395 Jul  4 11:45 .bashrc
```

看到了吗？属性改变了喔！由于一个档案有三组属性，所以你可以发现上面 777 为三组，而由于我们将所有的属性都打开，所以数字都相加，亦即『 r+w+x = 4+2+1 = 7』

那如果要将属性变成『 -rwxr-xr-- 』呢？那么就成为 [4+2+1][4+0+1][4+0+0]=754 啰！所以你需要下达 chmod 754 filename。 最常发生的一个问题就是，常常我们以 vi 编辑一个 shell 的文字文件后，他的属性通常是 -rw-rw-rw- 也就是 666 的属性，如果要将他变成可执行档，并且不要让其它人修改此一档案的话， 那么就需要 -rwxr-xr-x 这一个 755 的属性，所以 chmod 755 test.sh 就需要这样做啰！

另外，有些档案你不希望被其它人看到，例如 -rwxr-----，那么就下达 chmod 740 filename 吧！

例题三：将刚刚您的 .bashrc 这个档案的属性改回原来的 -rw-r--r-- 答： chmod 644 .bashrc

- 符号类型改变档案权限

还有一个改变属性的方法呦！从之前的介绍中我们可以发现，基本上就九个属性分别是(1)user (2)group (3)others 三群啦！那么我们就可以藉由 u, g, o 来代表三群的属性！此外， a 则代表 all 亦即全部的三群！那么读写的属性就可以写成了 r, w, x 啰！也就是可以使用底下的方式来看：

chmod	u g o a	+(加入) -(除去) =(设定)	r w x	档案或目录

来实作一下吧！假如我们要『设定』一个档案的属性为『-rwxr-xr-x』时，基本上就是：

- user (u)：具有可读、可写、可执行的权限；
- group 与 others (g/o)：具有可读与执行的权限。

所以就是：

```
[root@linux ~]# chmod  u=rwx,go=rx  .bashrc
# 注意喔！那个 u=rwx,go=rx 是连在一起的，中间并没有任何空格符！
[root@linux ~]# ls -al .bashrc
-rwxr-xr-x  1 root root 395 Jul  4 11:45 .bashrc
```

请注意， u=rwx,og=rx 这一段文字之间并没有空格符隔开呦！不要搞错啰！ 那么假如是『 -rwxr-xr-- 』? 可以使用『 chmod u=rwx,g=rx,o=r filename 』来设定。此外，如果我不知道原先的档案属性，而我只想要增加 .bashrc 这个档案的每个人均可写入的权限，那么我就可以使用：

```
[root@linux ~]# ls -al .bashrc
-rwxr-xr-x  1 root root 395 Jul  4 11:45 .bashrc
[root@linux ~]# chmod  a+w  .bashrc
[root@linux ~]# ls -al .bashrc
-rwxrwxrwx  1 root root 395 Jul  4 11:45 .bashrc
```

而如果是要将属性去掉而不更动其它的属性呢？！例如要拿掉所有人的 x 的属性，则：

```
[root@linux ~]# chmod  a-x  .bashrc
[root@linux ~]# ls -al .bashrc
-rw-rw-rw-  1 root root 395 Jul  4 11:45 .bashrc
```

知道 +, -, = 的不同点了吗？对啦！ + 与 - 的状态下，只要是没有指定到的项目， 则该属性『不会被变动』，例如上面的例子中，由于仅以 - 拿掉 x 则其它两个保持当时的值不变！呵呵！多多实作一下，

你就会知道如何改变属性啰！ 这在某些情况底下很好用的～举例来说，您想要教一个朋友如何让一个程序可以拥有执行的权限， 但您又不知道该档案原本的权限为何，此时，利用 chmod a+x filename ，就可以让该程序拥有执行的权限了。 是否很方便？

目录属性的意义：

刚刚上面我们提到的属性几乎都是针对一般档案的特性在说明，那么如果是针对目录时， 那个 r, w, x 对目录是什么意义呢？简单的说：

- r (read contents in directory)： 表示具有读取目录结构清单的权限，所以当您具有读取 (r) 一个目录的权限时， 您就可以利用 ls 这个指令将该目录的内容列表显示出来！
- w (modify contents of directory)： 这个可写入的权限对目录来说，是很了不起的！ 因为他表示您将具有异动该目录结构清单的权限，也就是底下这些权限：
    - 建立新的档案与目录；
    - 删除已经存在的档案与目录(不论该档案是属于谁的！)
    - 将已存在的档案或目录进行更名；
    - 搬移该目录内的档案、目录位置。

  所以说，如果您是一般身份使用者，例如鸟哥的账号 dmtsai ，那么在 /home/dmtsai 这个家目录内， 无论是谁 (包括 root) 建立的档案，无论该档案属于谁，无论该档案的属性是什么， dmtsai 这个使用者都『有权力将该档案删除』的喔！
- x (access directory)： 这个在上头我们已经稍微提过了，这个 x 与能否进入该目录有关呢！

好了，那么我们来简单的做个测试看看，底下可能会有很多您没有见过的指令， 不要担心，先照著作看看，等到未来提到该指令时，您自然就会了解了。

```
[root@linux ~]# cd /tmp
[root@linux tmp]# mkdir testing
[root@linux tmp]# chmod 744 testing
[root@linux tmp]# touch testing/testing
[root@linux tmp]# chmod 600 testing/testing
# 这个 mkdir 是在建立目录用的指令！是 make directory 的缩写。
# 我们用 root 的身份在 /tmp 底下建立一个名为 testing 的目录，
# 并且将该目录的权限变为 744 ，该目录的拥有者为 root 喔！
# 另外， touch 可以用来建立一个没有内容的档案，因此， touch testing/testing
# 可以建立一个空的 /tmp/testing/testing 档案喔！
[root@linux tmp]# ls -al
drwxr--r--   2 root root 4096 Jul 14 01:05 testing
# 仔细看一下，目录的权限是 744 ，且所属群组与使用者均是 root 喔！
# 接下来，我们将 root 的身份切换成为一般身份使用者。
# 鸟哥的系统里面有个 dmtsai 的一般身份使用者账号，所以切换身份成为 dmtsai
```

```
[root@linux tmp]# su dmtsai
# 那个 su 的指令是用来『变换身份』的一个指令，我们未来会详细介绍。
# 注意看，底下这一行中，发现使用者变为 dmtsai 了，而且提示字符变成 $ 了！
# 也就是说，现在操作系统的人变成 dmtsai 了！那么 dmtsai 这个人对于
# /tmp/testing 是属于 others 的权限，那他可以对 /tmp/testing 干嘛？
[dmtsai@linux tmp]$ ls -l testing   <== 此时身份为 dmtsai
total 0
?---------  ? ? ? ?           ? testing
# 可以查阅里面的信息喔！因为 dmtsai 具有 r 的权限，不过，毕竟权限不够，
# 很多资料竟然是问号 (?) 来的～怪怪的紧～
[dmtsai@linux tmp]$ cd testing  <== 此时身份为 dmtsai
bash: cd: testing/: Permission denied
# 发现了吗？即使我们具有 r 的权限，但是没有 x ，所以
# dmtsai 无法进入 /tmp/testing 喔！
[dmtsai@linux tmp]$ exit
[root@linux tmp]# chown dmtsai testing
# 使用 exit 就可以离开 su 的功能了。我们将这个 testing 目录的拥有者设定为
# dmtsai ，此时 dmtsai 就成为 owner 了，那么这个使用者又能干麻呢？
[root@linux tmp]# su dmtsai
[dmtsai@linux tmp]$ cd testing  <== 此时身份为 dmtsai
[dmtsai@linux testing]$ ls -l  <== 此时身份为 dmtsai
-rw-------  1 root root 0 Jul 14 01:13 testing
# 再切换身份成为 dmtsai ，此时就能够进入 testing 了！查阅一下内容。
# 发现了 testing 这个档案存在喔！权限是只有 root 才能够存取～
# 那我们测试一下能否删除呢？
[dmtsai@linux testing]$ rm testing  <== 此时身份为 dmtsai
rm: remove write-protected regular empty file `testing'? y
# 竟然可以删除！这样理解了吗？！
```

透过上面这个简单的步骤，您就可以清楚的知道， x 在目录当中是与『能否进入该目录』有关， 至于那个 w 则具有相当重要的权限，因为他可以让使用者删除、更新、新建档案或目录， 是个很重要的参数啊！这样可以理解了吗？！ ^_^

Linux 档案种类与附档名

我们在学习 Linux 之前，就跟大家灌输过一个概念，那就是，任何装置在 Linux 底下都是档案， 不仅如此，连数据沟通的接口也有专属的档案在负责～所以，您会了解到， Linux 的档案种类真的很多～ 除了前面提到的那个 -, d 亦即所谓的一般档案与目录档案之外，还有哪些种类的档案呢？

- 档案种类：

我们在刚刚的属性介绍中提到了最前面的标志 （ d 或 - ） 可以代表目录或档案，那就是不同的档案种类啦！Linux 的档案种类主要有底下这几种：

- 正规档案 (regular file)： 就是一般我们在进行存取的类型的档案，在由 ls －al 所显示出来的属性方面，第一个属性为 [－]，例如 [-rwxrwxrwx]。另外，依照档案的内容，又大略可以分为：

    - 纯文字文件(ASCII)：这是 Unix 系统中最多的一种档案类型啰，称为纯文字文件是因为内容为我们人类可以直接读到的数据， 例如数字、字母等等。几乎只要我们可以用来做为设定的档案都属于这一种档案类型。 举例来说，您可以下达『 cat ~/.bashrc 』就可以看到该档案的内容。 (cat 是将一个档案内容读出来的指令)

    - 二进制文件(binary)：还记得我们在『 Linux 是什么 』那一章里面的 GNU 发展史中提过， 我们的系统其实仅认识且可以执行二进制档案 (binary file) 吧？没错～ 您的 Linux 当中的可执行档 (scripts, 文字型批次文件不算) 就是这种格式的啦～ 举例来说，刚刚下达的指令 cat 就是一个 binary file 。

    - 数据格式文件(data)： 有些程序在运作的过程当中会读取某些特定格式的档案，那些特定格式的档案可以被称为数据文件 (data file)。举例来说，我们的 Linux 在使用者登入时，都会将登录的数据记录在 /var/log/wtmp 那个档案内，该档案是一个 data file ，他能够透过 last 这个指令读出来！ 但是使用 cat 时，会读出乱码～因为他是属于一种特殊格式的档案。瞭乎？

- 目录 (directory)：就是目录啰～第一个属性为 [d]，例如 [drwxrwxrwx]。

- 连结档 (link)：就是类似 Windows 底下的快捷方式啦！第一个属性为 [l]， 例如 [lrwxrwxrwx] ；

- 设备与装置文件 (device)：与系统周边及储存等相关的一些档案， 通常都集中在 /dev 这个目录之下！通常又分为两种：

    - 区块 (block) 设备档 ：就是一些储存数据， 以提供系统存取的接口设备，简单的说就是硬盘啦！例如你的一号硬盘的代码是 /dev/hda1 等等的档案啦！第一个属性为 [b]；

    - 字符 (character) 设备档 ：亦即是一些串行端口的接口设备， 例如键盘、鼠标等等！第一个属性为 [c]。

- 资料接口文件 (sockets)：既然被称为数据接口文件， 想当然尔，这种类型的档案通常被用在网络上的数据承接了。我们可以启动一个程序来监听客户端的要求， 而客户端就可以透过这个 socket 来进行数据的沟通了。第一个属性为 [s]， 最常在 /var/run 这个目录中看到这种档案类型了。

- 数据输送文件 (FIFO, pipe)： FIFO 也是一种特殊的档案类型，他主要的目的在解决多个程序同时存取一个档案所造成的错误问题。FIFO 是 first-in-first-out 的缩写。第一个属性为 [p] 。

那么使用刚刚的『 ls -al 』这个指令，你就可以简单的经由判断每一个档案的 第一个属性来了解这个档案是何种类型！很简单吧！ 除了设备文件是我们系统中很重要的档案，最好不要随意修改之外（通常他也不会让你修改的啦！）， 另一个比较有趣的档案就是连结档。如果你常常将应用程序捉到桌面来的话，你

就应该知道在 Windows 底下有所谓的『快捷方式』。同样的，你可以将 linux 下的连结档简单的视为一个档案或目录的快捷方式。至于 socket 与 FIFO 档案比较难理解，因为这两个咚咚与程序 (process) 比较有关系，这个等到未来您了解 process 之后，再回来查阅吧!此外，也可以透过 man fifo 及 man socket 来查阅系统上的说明!

- Linux 档案附档名:

基本上， Linux 的档案是没有所谓的『附档名』的，因为由前面的说明我们可以知道， 一个 Linux 档案能不能被执行，与他的第一栏的十个属性有关， 与文件名根本一点关系也没有。这个观念跟 Windows 的情况不相同喔! 在 Windows 底下，能被执行的档案附档名通常是 .com .exe .bat 等等，而在 Linux 底下， 只要你的属性当中有 x 的话，例如 [-rwx-r-xr-x] 即代表这个档案可以被执行喔!

不过，可以被执行跟可以执行成功是不一样的～举例来说，在 root 家目录下的 install.log 是一个纯文字文件，如果经由修改权限成为 -rwxrwxrwx 后，这个档案能够被执行吗? 当然不行～因为他的内容根本就没有可以执行的数据。所以说，这个 x 代表这个档案具有可执行的能力， 但是能不能执行成功，当然就得要看该档案的内容啰～

虽然附档名没有什么真的帮助，不过，由于我们仍然希望可以藉由附档名来了解该档案是什么东西?！所以， 通常我们还是会以适当的附档名来表示该档案是什么种类的。底下有数种常用的附档名:

- *.sh ： 批次檔 (scripts)，因为批次档为使用 shell 写成的，所以附档名就编成 .sh 啰;
- *Z, *.tar, *.tar.gz, *.zip, *.tgz： 经过打包的压缩档。这是因为压缩软件分别为 gunzip, tar 等等的，由于不同的压缩软件，而取其相关的附档名啰!
- *.html, *.php：网页相关档案，分别代表 HTML 语法与 PHP 语法的网页档案啰! .html 的档案可使用网页浏览器来直接开启，至于 .php 的档案， 则可以透过 client 端的浏览器来 server 端浏览，以得到运算后的网页结果呢!

另外，还有程序语言如 perl 的档案，其附档名也可能取成 .pl 这种档名! 基本上， Linux 上面的档名真的只是让你了解该档案可能的用途而已， 真正的执行与否仍然需要属性的规范才行!例如虽然有一个档案为可执行文件，如有名的代理服务器软件 squid ，不过，如果这个档案的属性被修改成无法执行时，那么他就变成不能执行啰! 这种问题最常发生在档案传送的过程中。例如你在网络上下载一个可执行档，但是偏偏在你的 Linux 系统中就是无法执行! 呵呵! 那么就是可能档案的属性被改变了! 不要怀疑，从网络上传送到你的 Linux 系统中，档案的属性确实是会被改变的喔!

再提个另外! 在 Linux 底下，每一个档案或目录的文件名最长可以到达 255 的字符，加上完整路径时，最长可达 4096 个字符，是相当长的档名喔!我们希望 Linux 的文件名称可以一看就知道该档案在干嘛的，所以档名通常是很长很长! 而用惯了 Windows 的人可能会受不了，因为文件名称通常真的都很长，对于用惯 Windows 而导致打字速度不快的朋友来说，嗯! 真的是很困扰.....不过，只得劝您好好的加强打字的训练啰! 而由前面一章的热键您也会知道，其实可以透过 [tab] 按键来确认档案的文件名的! 这很好用啊! 当然啦，如果您已经读完了本书第三篇关于 BASH 的用法，那么您将会发现 『哇! 变量真是一个相当好用的东西呐! 』 嗯! 看不懂，没关系，到第三篇谈到 bash 再说!

- Linux 文件名称的限制:

由于 Linux 在文字接口下的一些指令操作关系，一般来说，您在设定 Linux 底下的文件名称时， 最好可以避免一些特殊字符比较好！例如底下这些：

＊？＞＜；＆！［］｜＼’″｀（）｛｝

因为这些符号在文字接口下，是有特殊意义的！另外，文件名称的开头为小数点『.』时， 代表这个档案为『隐藏档』喔！同时，由于指令下达当中，常常会使用到 -option 之类的参数， 所以您最好也避免将档案档名的开头以 - 或 + 来命名啊！

Linux 目录配置

在了解了每个档案的相关种类与属性，以及了解了如何更改档案属性的相关信息后，再来要了解的就是，为什么每套 Linux distributions 他们的设定档啊、执行文件啊、每个目录内放置的咚咚啊，其实都差不多？原来是有一套标准依据的哩！我们底下就来瞧一瞧。

Linux 目录配置的依据 FHS

因为 Linux 的开发者实在太多了，如果每个人都发展出属于自己的目录配置方法， 那么将可能会造成很多管理上的困扰。您能想象，您进入一个企业之后，所接触到的 Linux 目录配置方法竟然跟您以前学的完全不同吗？！很难想象吧～所以，后来就有所谓的 Filesystem Hierarchy Standard (FHS) 标准的出炉了！

这个 FHS (http://www.pathname.com/fhs/) 事实上仅是规范出在根目录 (/) 底下各个主要的目录应该是要放置什么样的档案而已。 FHS 定义出两层规范出来，第一层是 / 底下的各个目录应该要放置什么样内容的档案数据，例如 /etc 应该要放置设定档， /bin 与 /sbin 则应该要放置可执行档等等。第二层则是针对 /usr 及 /var 这两个目录的次目录来定义的。例如 /var/log 放置系统登录文件、/usr/share 放置共享数据等等。

由于 FHS 仅是定义出最上层 (/) 及次层 (/usr, /var) 的目录内容应该要放置的档案数据， 因此，在其它个次目录层级内，就可以随开发者自行来配置了。举例来说， FC4 的网络设定数据放在 /etc/sysconfig/network-script/ 目录下，但是 SuSE Server 9 则是将网络放置在 /etc/sysconfig/network/ 目录下，目录名称可是不同的呢！

另外，在 Linux 底下，所有的档案与目录都是由根目录 / 开始的！那是所有目录与档案的源头～ 然后再一个一个的分支下来，有点像是树枝状啊～因此，我们也称这种目录配置方式为：『目录树 (directory tree)』 这个目录树有什么特性呢？他主要的特性有：

- 目录树的启始点为根目录 (/, root)；
- 每一个目录不止能使用本地端的 partition 的档案系统，也可以使用网络上的 filesystem 。举例来说， 可以利用 Network File System (NFS) 服务器挂载某特定目录等。
- 每一个档案在此目录树中的文件名(包含完整路径)都是独一无二的。

此外，根据档名写法的不同，也可将所谓的路径 (path) 定义为绝对路径 (absolute) 与相对路径 (relative)。 绝对路径为：由根目录 (/) 开始写起的文件名或目录名称， 例如 /home/dmtsai/.bashrc；相对路径为相对于目前路径的文件名写法。 例如 ./home/dmtsai 或 ../../home/dmtsai/ 等等。反正开头不是 / 就属于相对路径的写法， 而您必须要了解，相对路径是以『您当前所在路径的相对位置』来表示的。举例来说，您目前在 /home 这个目录下， 如果想要进入 /var/log 这个目录时，可以怎么写呢？

- cd /var/log (absolute)
- cd ../var/log (relative)

因为您在 /home 底下，所以要回到上一层 (../) 之后，才能继续往 /var 来移动的！ 特别注意这两个特殊的目录：

- . ：代表当前的目录，也可以使用 ./ 来表示；
- .. ：代表上一层目录，也可以 ../ 来代表。

这个 . 与 .. 目录概念是很重要的，您常常会看到 cd .. 或 ./command 之类的指令下达方式， 就是代表上一层与目前所在目录的工作状态喔！很重要的呐！此外，针对『档名』与『完整档名 (由 / 开始写起的文件名)』 的字符限制大小为：

- 单一档案或目录的最大容许文件名为 255 个字符；
- 包含完整路径名称及目录 (/) 之完整档名为 4096 个字符。

我们知道 /var/log/ 底下有个文件名为 message ，这个 message 档案的最大的档名可达 255 个字符。var 与 log 这两个上层目录最长也分别可达 255 个字符。但总的来说，由 /var/log/messages 这样完整档名最长则可达 4096 个字符。这样可以理解了吧！？ ^_^

Tips:
这个 root 在 Linux 里面的意义真的很多很多～多到让人搞不懂那是啥玩意儿。 如果以『账号』的角度来看，所谓的 root 指的是『系统管理员！』的身份， 如果以『目录』的角度来看，所谓的 root 意即指的是根目录，就是 / 啦～ 要特别留意喔！



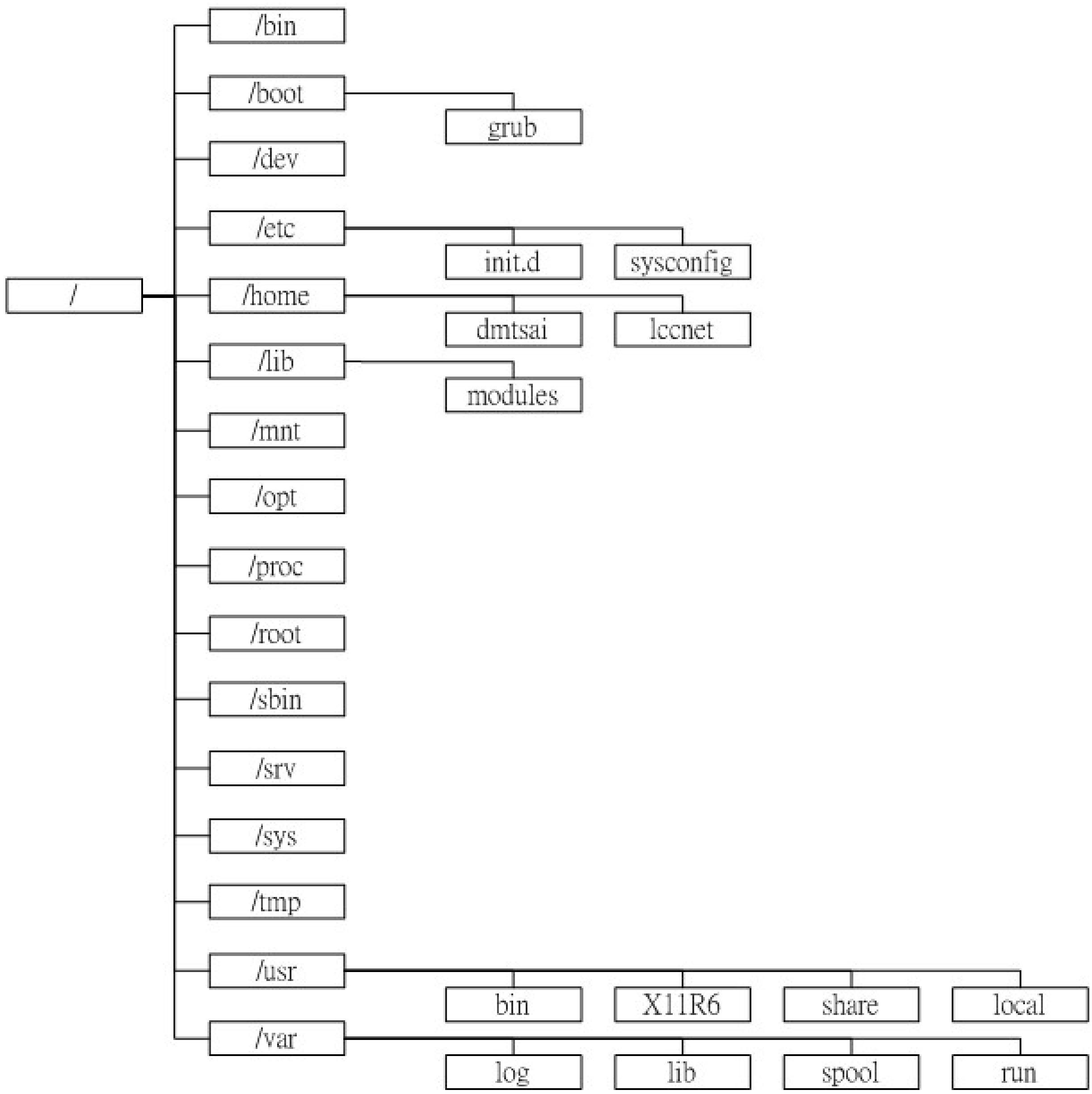
目录配置的内容

从前一小节的说明，您可以了解的是， FHS 定义出两层目录内的规范，那么如果您来到根目录查阅目录数据， 会显示什么呢？

```
[root@linux ~]# ls -l /
drwxr-xr-x   2 root root  4096 Jul 14 05:22 bin
drwxr-xr-x   3 root root  4096 Jul  9 05:18 boot
drwxr-xr-x   9 root root  4880 Jul 11 00:45 dev
drwxr-xr-x   6 root root  4096 Jun 29 01:06 disk1
drwxr-xr-x   3 root root  4096 Jun 25 08:53 disk2
drwxr-xr-x  83 root root 12288 Jul 14 05:23 etc
drwxr-xr-x   6 root root  4096 May 30 20:07 home
drwxr-xr-x  10 root root  4096 Jul 14 05:23 lib
drwx------   2 root root 16384 Jun 25 16:21 lost+found
drwxr-xr-x   3 root root  4096 Jun 25 19:34 media
drwxr-xr-x   2 root root  4096 Apr 25 23:54 misc
drwxr-xr-x   2 root root  4096 May 23 12:28 mnt
drwxr-xr-x   2 root root  4096 May 23 12:28 opt
```

```
dr-xr-xr-x  59 root root     0 Jul 10 01:25 proc
drwx------   9 root root  4096 Jul 13 11:31 root
drwxr-xr-x   2 root root  4096 Jul 14 05:22 sbin
drwxr-xr-x   2 root root  4096 Jun 25 08:23 selinux
drwxr-xr-x   2 root root  4096 May 23 12:28 srv
drwxr-xr-x  10 root root     0 Jul 10 01:25 sys
drwxr-xr-x  10 root root  4096 Jun 25 20:24 system
drwxrwxrwt  10 root root  4096 Jul 14 05:23 tmp
drwxr-xr-x  14 root root  4096 Jun 25 08:27 usr
drwxr-xr-x  24 root root  4096 Jun 25 20:16 var
```

从属性的角度来看，上面的档名每个都是『目录名称』，较为特殊的是 root ，由于 root 这个目录是管理员 root 的家目录，这个家目录可重要了！ 所以一定要设定成较为严密的 700 (rwx------)这个属性才行呐！如果以较为完整的树状目录来视察的话， 可以将整个 Linux 的树状目录绘制成下图：



图四、根据 FHS 定义的各层目录相关性

请注意，每个目录都是依附在 / 这个根目录底下的，所以我们在安装的时候一定要有一个 / 对应的 partition 才能安装的原因即在于此！这也就是我们俗称的『树状目录』啰！而根据 FHS 定义出来的每个目录内应该放置的档案内容为：

目录	应放置档案内容
/	根目录 root (/)，一般建议在根目录底下只接目录，不要直接有档案在 / 底下。根目录是开机的时候系统第一个挂载的 partition ，所以，所有开机过程会用到的档案， 应该都要放置在这个 partition 当中。举例来说， /etc, /bin, /dev, /lib, /sbin 这五个次目录都应该要与根目录连在一起，不可独立成为某个 partition 呢！

/bin, /usr/bin, /usr/local/bin	除了 /bin 之外， /usr/local/bin, /usr/bin 也是放置『使用者可执行的 binary file 的目录』喔！举例来说， ls, mv, rm, mkdir, rmdir, gzip, tar, cat, cp, mount 等等重要指令都放在这个目录当中。
/boot	这个目录主要的目的是放置 Linux 系统开机会用到的档案。开机会用到什么呢？没错～就是 Linux 的核心档案。这个目录底下文件名为 vmlinuz 的就是 Linux 的 Kernel 啦!粉重要的东西！而如果你的开机管理程序 (loader) 选择 grub 的话，那么这个目录内还有 /boot/grub 这个次目录呦！
/dev	在 Linux 系统上,任何装置与接口设备都是以档案的型态存在于这个目录当中的。您只要透过存取这个目录底下的某个档案，就等于存取某个装置啰～ 主要又分为接口设备 (character device),例如键盘、鼠标等;以及储存设备 (block device),例如硬盘、光盘等等。在此目录下的档案会多出两个属性，分别是 major device number ， 与 minor device number 。我们的系统核心就是透过这两个 number 来判断装置的呢！ 比要重要的档案有 /dev/null, /dev/tty[1-6], /dev/ttyS*, /dev/lp*, /dev/hd*, /dev/sd* 等等
/etc	系统主要的设定文件几乎都放置在这个目录内，例如人员的账号密码文件、 各种服务的启始档等等。一般来说，这个目录下的各档案属性是可以让一般使用者查阅的， 但是只有 root 有权力修改。并且在此目录下的档案几乎都是 ASCII 的纯文本文件哩。 不过， FHS 建议不要放置可执行文件在这个目录中喔。比较重要的档案有： /etc/inittab, /etc/init.d/, /etc/modprobe.conf, /etc/X11, /etc/fstab, /etc/sysconfig/ 等等。另外，其下重要的目录有： • /etc/init.d/：所有服务的预设启动 script 都是放在这里的，例如要启动或者关闭 iptables 的话： /etc/init.d/iptables start /etc/init.d/iptables stop • /etc/xinetd.d/：这就是所谓的 super daemon 管理的各项服务的设定文件目录。 • /etc/X11：与 X Window 有关的各种设定档都在这里，尤其是 xorg.conf 或 XF86Config 这两个 X Server 的设定档。
/home	这是系统预设的使用者家目录 (home directory)。在你新增一个一般使用者账号时， 预设的使用者家目录都会规范到这里来。比较重要的是，家目录有两种代号喔： ~：代表目前这个使用者的家目录，而 ~dmtsai ：则代表 dmtsai 的家目录！
/lib, /usr/lib, /usr/local/lib	系统会使用到的函式库放置的目录。 程序在运作的过程中，可能会呼叫一些额外的功能参数，那需要函式库的协助！ 这些函式库就放在此处。比较重要的是 /lib/modules 这个目录内会摆放 kernel 的相关模块喔！
/lost+found	系统不正常产生错误时，会将一些遗失的片段放置于此目录下， 通常这个目录会自动出现在某个 partition 最顶层的目录下。例如你加装一棵硬盘于 /disk 中，那在这个目录下就会自动产生一个这样的目录 /disk/lost+found

/mnt /media	这是软盘与光盘预设挂载点的地方；通常软盘挂在 /mnt/floppy 下，而光盘挂在 /mnt/cdrom 下，不过也不一定啦！只要你高兴，随便找一个地方来挂载也可以呀！另外，目前也规划出另一个 /media 的目录呢！与 /mnt 有点类似啦～
/opt	这是给主机额外安装软件所摆放的目录。举例来说， FC4 使用的是 Fedora 社群开发的软件， 如果您今天想要自行安装新的 KDE 桌面软件的话，可以将该软件安装在这个目录下的意思。 不过，以前的 Linux 系统中，我们还是习惯放置在 /usr/local 目录下呢！
/proc	这个目录本身是一个『虚拟档案系统』喔！他放置的数据都是在内存当中， 例如系统核心、形成信息、接口设备的状态及网络状态等等。因为这个目录下的数据都是在内存当中， 所以本身不占任何硬盘空间啊！比较重要的档案例如：/proc/cpuinfo, /proc/dma, /proc/interrupts, /proc/ioports, /proc/net/* 等等。
/root	系统管理员 (root) 的家目录。之所以放在这里， 是因为我们提过，系统第一个开机就被挂载的 partition 为 / ， 而我们希望 /root 能够与 / 放在同一块 partition 上面之故。
/sbin, /usr/sbin, /usr/local/sbin	放置一些系统管理员才会动用到的执行指令，例如： fdisk, mke2fs, fsck, mkswap, mount 等等。与 /bin 不太一样的地方，这几个目录是给 root 等系统管理用的。但是本目录下的执行文件还是可以让一般使用者用来『察看』而不能设定喔！
/srv	一些服务启动之后，这些服务所需要取用的数据目录。举例来说，WWW 服务器需要的网页资料就可以放置在 /srv/www 里面。
/tmp	这是让一般使用者或者是正在执行的程序暂时放置档案的地方。 这个目录是任何人都能够存取的，所以您需要定期的清理一下。当然，重要数据不可放置在此目录啊！
/usr	由 FHS 规范的第二层内容，在 /usr 此目录下，包含系统的主要程序、 图形接口所需要的档案、额外的函式库、本机端所自行安装的软件，以及共享的目录与文件等等， 都可以在这个目录当中发现。事实上，他有点像是 Windows 操作系统当中的『Program files』与 『WinNT』这两个目录的结合！在此目录下的重要次目录有： • /usr/bin, /usr/sbin：一般身份使用者与系统管理员可执行的档案放置目录； • /usr/include：c/c++等程序语言的档头 (header) 与包含档(include) 放置处， 当我们以 tarball 方式 (*.tar.gz 的方式安装软件) 安装某些数据时，会使用到里头的许多包含档喔！； • /usr/lib：各应用软件的函式库档案放置目录； • /usr/local:本机端自行安装的软件预设放置的目录。目前也适用于 /opt 目录。 在你安装完了 Linux 之后，基本上所有的配备你都有了，但是软件总是可以升级的， 例如你要升级你的 proxy 服务，则通常软件预设的安装地方就是在 /usr/local (local 是『当地』的意思)，同时，安装完毕之后所得到的执行文件，为了与系统原先的执行文件有分别， 因此升

	级后的执行档通常摆在 /usr/local/bin 这个地方。给个建议啦， 通常鸟哥都会将后来才安装上去的软件放置在这里，因为便于管理呦； • /usr/share：共享文件放置的目录，例如底下两个目录： • /usr/share/doc：放置一些系统说明文件的地方，例如你安装了 grub 了，那么在该目录底下找一找，就可以查到 lilo 的说明文件了！很是便利！ • /usr/share/man：manpage 的文件档案目录；那是什么？呵呵！ 就是你使用 man 的时候，会去查询的路径呀！例如你使用 man ls 这个指令时，就会查出 /usr/share/man/man1/ls.1.gz 这个说明档的内容啰！ • /usr/src：Linux 系统相关的程序代码放置目录，例如 /usr/src/linux 为核心原始码！ • /usr/X11R6：系统内的 X Window System 所需的执行档几乎都放在这！
/var	这个目录也很重要，也是 FHS 规范的第二层目录内容。他主要放置的是针对系统执行过程中，常态性变动的档案放置的目录。举例来说，例如快取档案 (cache) 或者是随时变更的登录档 (log file) 都是放在这个目录中的。此外，某些软件执行过程中会写入的数据库档案， 例如 MySQL 数据库，也都写入在这个目录中！很重要吧！他底下的重要目录有： • /var/cache：程序档案在运作过程当中的一些暂存盘； • /var/lib：程序本身执行的过程中，需要使用到的数据文件放置的目录，举例来说， locate 这个数据库与 MySQL 及 rpm 等数据库系统，都写在这个目录内。 • /var/log：登录文件放置的目录。很重要啊！例如 /var/log/messages 就是总管所有登录档的一个档案！ • /var/lock：某些装置具有一次性写入的特性，例如 tab (磁带机)， 此时，为了担心被其它人干扰而破坏正在运作的动作，因此，会将该装置 lock (锁住)起来， 以确定该装置只能被单一个程序所使用啊！ • /var/run：某些程序或者是服务启动后，会将他们的 PID 放置在这个目录下喔！ • /var/spool：是一些队列数据存放的地方。举例来说，主机收到电子邮件后， 就会放置到 /var/spool/mail 当中，若信件暂时发不出去，就会放置到 /var/spool/mqueue 目录下， 使用者工作排程 (cron) 则是放置在 /var/spool/cron 当中！

需要特别注意的目录

在上一节当中我们大约了解了各个目录下所放置的档案的用途。或许您会看得很吃力，不过，不要担心， 等到后面的章节看完后，再回来这里瞧一瞧，就会很清楚的知道每个目录的意义啰！而这些目录当中， 有几个比较有趣的目录还是需要特别来提醒一下的：

- 建议不可与 root partition 分开的目录

我们在 安装 FC4 那个章节里面，有提到磁盘分割 (partition) 的概念对吧！在 Linux 的安装里面，最重要的就是根目录 / 所在的那个 partition 了。 我们也可以将其它的例如 /home 放在不同的

partition 里面。那么是否有『一定』要放在 root partition 内的目录呢？有啊！那就是：/etc/, /sbin/, /bin/, /dev/ 以及 /lib/ 这几个目录了。

为什么呢？因为我们的 Linux 系统在开机的时候，一开始进行核心加载时， 只会挂载一个 partition ，那就是 / 。但是开机的时候会用到很多的指令与函式库，举例来说，要挂载， 就得需要 mount 这支程序，而且我们也需要 init 这支程序，还需要用到很多的设定档，例如 /etc/inittab 等等。而我们核心的模块则是放置在 /lib 里面。当然， /dev 是所有装置放置的目录， 也需要在开机的时候使用到的。因此，这些目录都需要跟 / 绑在一起喔！ 先有概念即可，下面两章会跟您介绍所谓的 partition 概念的！

- 建议最好独立成为单一 partition 的目录

上面提到的是最好不要跟 / 这个 partition 分离的目录。至于有些目录则是因为安全性与特殊功能性，而希望能够独立成为一个自己的 partition 呢！例如： /home, /usr, /var, /tmp 等等。

我们必须要先知道的，系统上的使用者个人家目录在 /home 里面， 这个目录也是可能被使用的最频繁的目录之一。此外，为了资源分配较为平均， 我们可能会希望针对每个人限制他可以使用的最大硬盘总量 (quota)，在这个前提之下， 您就必须要将 /home 独立出来，而且最好这个 partition 能够大一点，尤其是您的 Linux 是作为档案服务器 (file server) 时，就更形重要了。

至于 /usr 则是一些程序安装的目录，也可以独立出来的；还有 /var/， 这个目录由于记录了相当多的常用数据，读取真的是很频繁，所以是『很容易挂点的 partition 一!』如果能够将他独立出来，那么当 /var/ 真的、万一、不小心挂点时，就不会影响到其它的 partition ， 最起码能有一定程度的安全性啦！

- 特别重要的几个目录

除了针对 partition 的观念来谈目录的重要性之外，有几个比较重要的目录您也需要了解一下：

- /etc：这个目录是系统设定文件放置的地方， 包括您系统上的账号与密码 (/etc/passwd, /etc/shadow)，还有开机时所要用到的各项设定值 (/etc/sysconfig/*) ，还有各主要的网络服务的设定文件，都在这个目录中。 意思就是说，如果这个目录底下的档案被删除或者是死掉了，嘿嘿～ 您的系统大概也就需要『很花功夫』的重建了～ ^_^。因此， 一般鸟哥都会定期将这个目录的所有档案给他备份下来， 反正这个目录的大小应该不会超过 50MB 才对，多多备份，有备无患啊！

- /usr/local：虽然说目前已经将这个目录的重要性移动到 /opt 了， 但是鸟哥还是比较习惯将我自己开发或自行额外安装的软件放置在这个 /usr/local 目录下。 如果您的 Linux 系统是多人共管的话，那么，养成一个良好的操作习惯是有必要的。 那么安装软件的习惯也要好好建立起来啊～不要随意安装呢！ 统一放置在 /usr/local 或者是 /opt 底下吧！ ^_^

- /var：在上面提过一次，这里再次强调。 这个目录是在管理系统运作过程中的重要中间暂存数据的，例如 /var/lib 与 /var/run 。 此外，最终的数据例如邮件 /var/spool/mail 也是放置在这个目录中～ 另外，几乎所有服务的登录文件 (可以记录谁、什么时候、由哪里登入主机、做了什么事等等信息！) 都放在 /var/log 这个目录下，因此，这个目录也很重要。记得常常去检查 /var/log/messages 这个档案是否有异常啊～

一般主机 partition 与目录的配置情况

好了，知道了 Linux 的档案权限，目前也知道了各个档案内可能摆放的数据是什么了， 那么再来说说你的目录与磁盘分割之间的相关性。通常一般的大型主机都不会将所有的数据放置在一个磁盘中 (就是只有一个『 / 』根目录)，这有几个目的：

- 安全性考虑：
  你的系统通常是在 /usr/ 中，而个人数据则可能放置在 /home 当中，至于一些开机数据则放置在 /etc 当中。如果将所有的数据放在一起，当你的系统不小心被黑客破坏，或者不小心自己砍了一个小东西， 则所有的咚咚也都跟着不见了.....这对于我们市井小民或许无所谓，再安装一次就好了， 但是对于一些大型企业可不行这样！因此需要将数据分别放置于不同的磁盘中，会比较保险些。

- 便利性：
  如果你需要升级你的系统的话，是否需要重新 format 安装呢？有些数据例如 /home 里面的数据为个人用户的数据，似乎与系统无关！所以如果你将这些数据分别放置于不同的磁盘， 则你要升级或者进行一些系统更动时，将比较有弹性。

你或许可以将你的系统做成这样的 partition 分布：

/
/boot
/usr
/home
/var

这是比较常见的磁盘分布情况，其中：

- / 根目录可以分配约 1 GB 以内;
- /boot 大概在 50 MB 就可以了，因为开机档案并不大;
- /var 就至少需要 1GB 以上，因为你的 mail 、 proxy 预设的储存区都在这个目录中，除非你要将一些设定改变！
- /home 与 /usr 通常是最大的，因为你所安装的数据都是在 /usr/ 当中，而用户数据则放置在 /home 当中，因此通常大家都会建议你将所剩下的磁盘空间平均分配给这两个目录说！ 不过也不一定啦！ /usr 大概给个 10G 就很多了～其它的可以都给 /home ，也可以保留一些剩余空间来作为以后的安装与设定用啊！

无论如何，每部主机的环境与功能用途都不相同，自然其磁盘的分配就会不太一样，因此， 上面的设定您就看看就好，等您将整本书或者整个网页内容全 K 完了，那么大概就知道怎样设定您的主机最恰当啦！

Linux 支持的档案系统

我们在本章前面提到的都是单一档案的属性与相关信息，以及单一目录在 Filesystem Hierarchy Standard (FHS) 当中的定义，还没有提到所谓的 partition 的相关概念。底下我们就约略来谈一谈，那么我们前面提到的这些档案、目录， 是放在什么样的档案系统内呢？

什么是档案系统 (filesystem) 呢？目前的操作系统大多数是将数据由硬盘读出来的，那么每个操作系统使用的硬盘在 x86 架构上的，都一样啊！都是同样的硬盘。 但是，每种操作系统都有其独特的读取档案的方法，也就是说， 每种操作系统对硬盘读取的方法不同，所以就造就了不同的档案系统了。

举例来说， Windows 98 预设的档案系统是 FAT (或 FAT16) 档案系统， Windows 2000 有所谓的 NTFS 档案系统，至于 Linux 的正统档案系统则为 ext2 (Linux second extended file system, ext2fs) 这一个。 我们的系统能不能读取某个档案系统，与前面提过的『核心功能』有关。 Linux 核心必须要能够认识某种档案系统后，我们的 Linux 才能读取该档案系统的数据内容啊！ 也就是说，你必须要将你所想要支持的档案系统编译到你的核心当中才能被支持 。因此，您可以发现， Windows 与 Linux 安装在同一个硬盘的不同 partition 时， Windows 将不能取用 Linux 的硬盘数据， Why？就因为 Windows 的核心不认识 Linux 的档案系统呀！

目前 Fedora Core IV 预设的档案系统为 ext3 (Third Extended File System)， 他是 Ext2 的升级版，主要是增加了日志 (journaling) 的功能，但是 ext3 还是向下支持 ext2 等。 另外，如果你需要将你原有的 Windows 系统也挂载在 Linux 底下的话，那么 Linux 同时也支持 MS-DOS, VFAT, FAT, BSD 等等的档案系统。至于 Window NT 的 NTFS 档案系统则不见得每一个 Linux distribution 都有支持，例如我们的 FC4 预设就没有支持了。 问我怎么看出来的？呵呵！ Linux 能够支持的档案系统与核心是否有编译进去有关， 所以你可以到你的 Linux 系统的：

```
/lib/modules/`uname -r`/kernel/fs
```

该目录底下看一看，如果有你想要的档案系统，那么这个核心就有支持啦！ 很多 Linux 所需要的功能都可以在 ext2 上面完成，不过 ext2 缺乏日志管理系统， 如果发生问题时，修复过程会比较慢一些。所以最近释出的 Linux distribution 大多已经预设采用 ext3 或 reiserfs 这种具有日志式管理的档案系统了。

那么什么是日志式档案系统呢？举例来说， ext3 与 ext2 有啥不同？ ext3 其实只是多做了一个日志式数据的纪录。当我们要在将数据写入硬盘时， ext2 是直接将数据写入，但是 ext3 则会将这个『要开始写入』的讯息写入日志式记录区， 然后才开始进行数据的写入。在数据写入完毕后，又将『完成写入动作』的讯息写入日志式记录区， 这有什么好处呢？最大的好处就是数据的完整性与『恢复力』。

什么是『恢复力』呢？早期的 ext2 档案系统如果发生类似断电后时，档案系统就得要检查档案一致性。 这个检查的过程要将整个 partition 内的档案做一个完整的比较，哇!很慢很慢、很久很久啊～ 如果是 ext3 的话，那么只要透过检查『日志记录区』就可以知道断电时，是否有哪些档案正在进行写入的动作， 只要检查这些地方即可～这样就能够节省很多档案检查的时间呢！

这样知道为何要选择 ext3 了吧？我们还可以引用 Red Hat 公司中，首席核心开发者 Michael K. Johnson 的话：

> 『为什么你想要从 ext 2 转换到 ext3 呢？有四个主要的理由：可利用性、数据完整性、速度及易于转换』『可利用性』，他指出，这意味着从系统中止到快速重新复原而不是持续的让 e2fsck 执行长时间的修复。ext3 的日志式条件可以避免数据毁损的可能。他也指出：『除了写入若干数据超过一次时，ext3 往往会较快于 ext2，因为 ext3 的日志使硬盘读取头的移动能更有效的进行』然而或许决定的因素还是在 Johnson 先生的第四个理由中。
>
> 『它是可以轻易的从 ext2 变更到 ext3 来获得一个强而有力的日志式档案系统而不需要重新做格式化』。『那是正确的，为了体验一下 ext3 的好处是不需要去做一种长时间的，冗长乏味的且易于产生错误的备份工作及重新格式化的动作』。

上列资料可在 Whitepaper: Red Hat's New Journaling File System: ext3 (http://www.redhat.com/support/wpapers/redhat/ext3/) 查阅得到。 所以啰，使用 ext3 或者是其它的日志式档案系统是有好处的，最大的好处当然是错误问题的排除效率比较高。 无论如何，您只要先晓得 ext2 是 Linux 正规的档案系统，而近年来的 ext3 等日志式档案系统则有取代的趋势。

- Linux 的 VFS (Virtual Filesystem Switch)：

了解了我们使用的档案系统之后，再来则是要提到，那么 Linux 的核心又是如何管理这些认识的档案系统呢？ 其实，整个 Linux 的系统都是透过一个名为 Virtual Filesystem Switch 的核心功能去读取 filesystem 的。 也就是说，整个 Linux 认识的 filesystem 其实都是 VFS 在进行管理， 我们使用者并不需要知道每个 partition 上头的 filesystem 是什么～ VFS 会主动的帮我们做好读取的动作呢～

这无疑是个很好用的功能～为什么呢？因为只要系统管理员一开始就设定好各主要 filesystem 对应的档案系统模块后， 核心的 VFS 就会主动接管该 filesystem 的存取模式。使用者可以在不晓得每个 filesystem 是什么的情况下， 就能自由的运用系统上的各种 filesystem 。很方便～不是吗？！ ^_^

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 早期的 Unix 系统文件名最多允许 14 个字符，而新的 Unix 与 Linux 系统中，文件名最多可以容许几个字符？

  单一文件名可达 255 字符，完整文件名 (包含路径) 可达 4096 个字符

- 当一个档案属性为 －rwxrwxrwx 则表示这个档案的意义为？

  任何人皆可读取、可写入亦可删除。

- 我需要将一个档案的属性改为 －rwxr-xr-- 请问该如何下达指令？

  chmod 754 filename 或 chmod u=rwx,g=rx,o=r filename

- 若我需要更改一个档案的拥有者与群组，该用什么指令？

  chown, chgrp

- Linux 传统的档案系统为何？此外，常用的 Journaling 档案格式有哪些？

  传统档案格式为：ext2,
  Journaling 有 ext3 及 Reiserfs 等

- 请问底下的目录与主要放置什么数据：
  /etc/, /etc/init.d, /boot, /usr/bin, /bin, /usr/sbin, /sbin, /dev, /var/log
  - /etc/：几乎系统的所有设定档案均在此，尤其 passwd,shadow
  - /etc/init.d：系统开机的时候加载服务的 scripts 的摆放地点

    - /boot：开机设定档，也是预设摆放核心 vmlinuz 的地方
    - /usr/bin, /bin：一般执行档摆放的地方
    - /usr/sbin, /sbin：系统管理员常用指令集
    - /dev：摆放所有系统装置档案的目录
    - /var/log：摆放系统登录档案的地方
- 若一个档案的档名开头为『 . 』，例如 .bashrc 这个档案，代表什么？另外，如何显示出这个文件名与他的相关属性？

  有『 . 』为开头的为隐藏档，需要使用 ls －a 这个 －a 的参数才能显示出隐藏档案的内容，而使用 ls －al 才能显示出属性。

参考数据

- 关于 Journaling 日志式文章的相关说明 http://www.linuxplanet.com/linuxplanet/reports/3726/1/

Linux 磁盘与档案系统管理

最近更新日期：2006/06/28

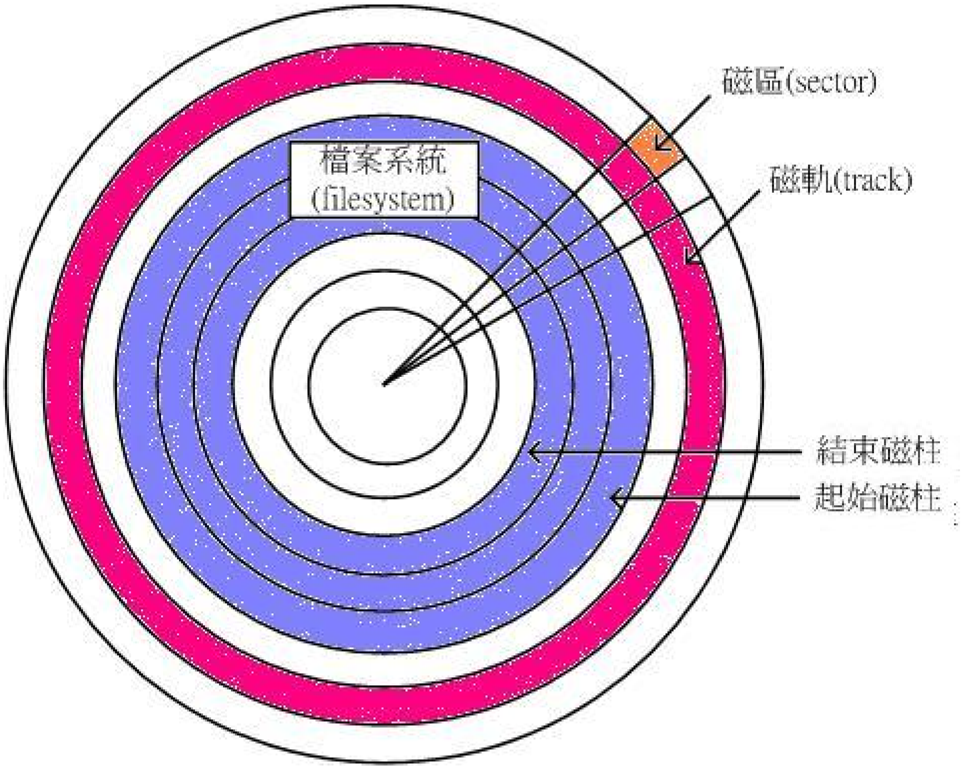
我们在前面的档案权限介绍的章节当中，提到很多的权限与属性的观念，那么接下来要了解的是， 这些属性是记录在硬盘的那个地方？这里就要特别了解到 Linux 档案系统(filesystem)是如何记录档案， 与档案是如何被读取的啰！而要了解整个档案系统的观念，就不能不知道硬盘的组成组件！ 所以，在这个章节当中，我们由最基础的硬盘组成组件介绍起，并介绍 inode 与连结文件等基本知识， 以及如何利用开机即可挂载的方式来使我们的各个 partition 可以在开机时就已经进行好挂载的动作喔！

认识 EXT2 档案系统

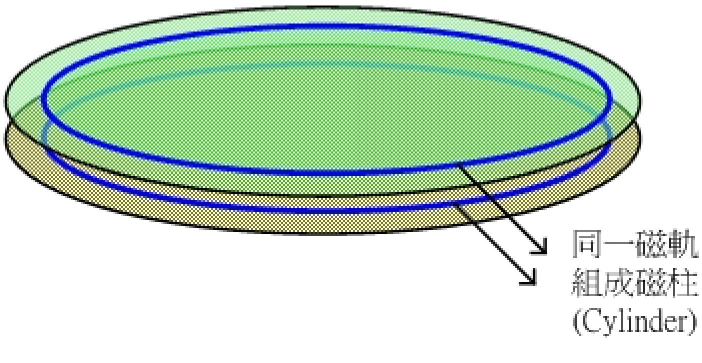
既然这个章节主要在探讨 Linux 的磁盘档案系统，所以我们当然就需要先来了解一下硬盘是个什么东西啦！ 首先，我们就来看一看硬盘的物理组成，了解了物理组成之后，再来说明一下怎么样进行硬盘的分割 (partition) 吧！

硬盘物理组成：

就硬盘的物理组件来说，硬盘其实是由许许多多的圆形硬盘盘所组成的， 依据硬盘盘能够容纳的数据量，而有所谓的单碟 (一块硬盘里面只有一个硬盘盘) 或者是多碟 (一块硬盘里面含有多个硬盘盘)的硬盘。在这里我们以单一个硬盘盘来说明，硬盘盘可由底下的图形来示意：



图一、硬盘盘示意图



图二、磁柱示意图

首先，硬盘里面一定会有所谓的磁头 (Head) 在进行该硬盘盘上面的读写动作，而磁头是固定在机械手臂上面的，机械手臂上有多个磁头可以进行读取的动作。 而当磁头固定不动 (假设机械手臂不动) ，硬盘盘转一圈所画出来的圆就是所谓的磁道(Track)；而如同我们前面刚刚提到的，一块硬盘里面可能具有多个硬盘盘， 所有硬盘盘上面相同半径的那一个磁道就组成了所谓的磁柱(Cylinder)。

例如上图二所示意，在两个硬盘盘上面的同一个磁道就是一个磁柱啦！这个磁柱也是磁盘分割(partition)

时的最小单位了； 另外，由圆心向外划直线，则可将磁道再细分为一个一个的扇区(Sector)，这个扇区就是硬盘盘上面的最小储存物理量了！ 通常一个 sector 的大小约为 512 Bytes 。以上就是整个硬盘的基本组件。

在计算整个硬盘的储存量时，简单的计算公式就是：Cylinder x Head x Sector x 512 Bytes。另外，硬盘在读取时，主要是『硬盘盘会转动， 利用机械手臂将磁头移动到正确的数据位置(单方向的前后移动)，然后将数据依序读出。』 在这个操作的过程当中，由于机械手臂上的磁头与硬盘盘的接触是很细微的空间，如果有抖动或者是脏污在磁头与硬盘盘之间时，就会造成数据的损毁或者是实体硬盘整个损毁～

因此，正确的使用计算机的方式，应该是在计算机通电之后，就绝对不要移动主机，并免抖动到硬盘， 而导致整个硬盘数据发生问题啊！另外，也不要随便将插头拔掉就以为是顺利关机！ 因为机械手臂必须要归回原位，所以使用操作系统的正常关机方式，才能够有比较好的硬盘保养啊！ 因为他会让硬盘的机械手臂归回原位啊！

磁盘分割 (Partition)：

在了解了硬盘的物理组件之后，再接着下来介绍的就是硬盘的分割(Partition)啰！ 为什么要进行硬盘分割啊？！因为我们必须要告诉操作系统：『 我这块硬盘可以存取的区域是由 A 磁柱到 B 磁柱』，如此一来，操作系统才能够控制硬盘磁头去 A-B 范围内的磁柱存取数据;如果没有告诉操作系统这个信息，那么操作系统就无法利用我们的硬盘来进行数据的存取了， 因为操作系统将无法知道他要去哪里读取数据啊！这就是磁盘分割(Partition)的重点了： 也就是记录每一个分割区(Partition)的起始与结束磁柱！

好了，那么这个分割区的起始与结束磁柱的数据放在哪里呢？！那就是我们在 Linux 安装与多重开机技巧 那个章节提到的 主要开机扇区(Master Boot Recorder, MBR)啰！事实上， MBR 就是在一块硬盘的第零轨上面，这也是计算机开机之后要去利用该硬盘时， 必须要读取的第一个区域！在这个区域内记录的就是硬盘里面的所有分割信息， 以及开机的时候可以进行开机管理程序的写入的处所啊！所以，当一个硬盘的 MBR 坏掉时，由于分割的数据不见了，呵呵，那么这个硬盘也就几乎可以说是寿终正寝了， 因为操作系统不知道该去哪个磁柱上读取数据啊～～

那么 MBR 有什么限制呢？他最大的限制来自于他的大小不够大到储存所有分割与开机管理程序的信息，因此,MBR 仅提供最多四个 partition 的记忆,这就是所谓的 Primary (P)与 Extended (E) 的 partition 最多只能有四个的原因了。所以说，如果你预计分割超过 4 个 partition 的话，那么势必需要使用 3P + 1E ，并且将所有的剩余空间都拨给 Extended 才行(记得呦！ Extended 最多只能有一个)，否则只要 3P + E 之后还有剩下的空间， 那么那些容量将成为废物而浪费了，所以结论就是『 如果您要分割硬盘时，并且已经预计规划使用掉 MBR 所提供的 4 个 partition (3P + E 或 4P)那么磁盘的全部容量需要使用光，否则剩下的容量也不能再被使用』。 不过，如果您仅是分割出 1P + 1E 的话，那么剩下的空间就还能再分割两个 primary partition ！

档案系统：

在告知系统我的 partition 所在的起始与结束磁柱之后，再来则是需要将 partition 格式化为『我的操作系统认识的档案系统(Filesystem)』啰!因为每个操作系统认识的 filesystem 并不相同!例如 Windows 操作系统在预设状态下就无法认识 Linux 的档案系统 (这里指 Linux 的标准档案系统 ext2)。所以当

然要针对我们的操作系统来格式化 partition 啰!

我们可以说，每一个 partition 就是一个 Filesystem ，那么一个 partition 是否可以具有两个 Filesystem 呢？！理论上应该是不行的！因为每个档案系统都有其独特的支持方式，例如 Linux 的 ext3 就无法被 Windows 系统所读取！而你将一个 partition 格式化的时候，总不能格式化为 ext3 也同时格式化为 fat32 吧？！那是不可能的啊！

不论是哪一种 filesystem ，数据总是需要储存的吧！既然硬盘是用来储存数据的，想当然尔， 数据就必须写入硬盘啦！刚刚我们提到硬盘的最小储存单位是 sector ，不过数据所储存的最小单位并不是 sector 喔，因为用 sector 来储存太没有效率了。怎么说呢？因为一个 sector 只有 512 Bytes ，而磁头是一个一个 sector 的读取，也就是说，如果我的档案有 10 MBytes ，那么为了读这个档案， 我的磁头必须要进行读取 (I/O) 20480 次！

为了克服这个效率上的困扰，所以就有逻辑区块(Block)的产生了！ 逻辑区块是在 partition 进行 filesystem 的格式化时， 所指定的『最小储存单位』，这个最小储存单位当然是架构在 sector 的大小上面(因为 sector 为硬盘的最小物理储存单位啊！)，所以啦， Block 的大小为 sector 的 2 的次方倍数。此时，磁头一次可以读取一个 block ，如果假设我们在格式化的时候，指定 Block 为 4 KBytes (亦即由连续的八个 sector 所构成一个 block)，那么同样一个 10 MBytes 的档案， 磁头要读取的次数则大幅降为 2560 次，这个时候可就大大的增加档案的读取效能啦！

不过，Block 单位的规划并不是越大越好喔！怎么说呢？因为一个 Block 最多仅能容纳一个档案 (这里指 Linux 的 ext2 档案系统)！这有什么问题呢？举例来说好了，假如您的 Block 规划为 4 KBytes ，而您有一个档案大小为 0.1 KBytes ，这个小档案将占用掉一个 Block 的空间，也就是说，该 Block 虽然可以容纳 4 Kbytes 的容量，然而由于档案只占用了 0.1 Kbytes ，所以，实际上剩下的 3.9 KBytes 是不能再被使用了，所以，在考虑 Block 的规划时，需要同时考虑到：

- 档案读取的效能
- 档案大小可能造成的硬盘空间浪费

因此，在规划您的磁盘时，需要留意到您主机的用途来进行规划较佳！例如 BBS 主机由于文章较短， 也就是说档案较小，那么 Block 小一点的好；而如果您的主机主要用在储存大容量的档案， 那么考虑到效能，当然 Block 理论上，规划的大一点会比较妥当啦！

Superblock：如同前面说的，当我们在进行磁盘分割(partition)时，每个磁盘分割槽(partition)就是一个档案系统(filesystem)， 而每个档案系统开始的位置的那个 block 就称为 superblock ，superblock 的作用是储存像是档案系统的大小、空的和填满的区块，以及他各自的总数和其它诸如此类的信息等等， 这也就是说，当您要使用这一个磁盘分割槽(或者说是档案系统)来进行数据存取的时候，第一个要经过的就是 superblock 这个区块了，所以啰， superblock 坏了，您的这个磁盘槽大概也就回天乏术了！

Linux 的 EXT2 档案系统(inode)：

看完了上面的说明，您应该对于硬盘有一定程度的认识了！好了，那么接下来就是要谈一谈 Linux 的档案系统(Filesystem)啰!我们这里以 Linux 最标准的 ext2 这个档案系统来作为说明。还记得我们在 Linux

档案属性与目录配置 那个章节提到的，在 Linux 系统当中，每个档案不止有档案的内容数据，还包括档案的种种属性，例如：所属群组、 所属使用者、能否执行、档案建立时间、档案特殊属性等等。由于 Linux 操作系统是一个多人多任务的环境，为了要保护每个使用者所拥有数据的隐密性， 所以具有多样化的档案属性是在所难免的！在标准的 ext2 档案系统当中，我们将每个档案的内容分为两个部分来储存，一个是档案的属性，另一个则是档案的内容。

为了应付这两个不同的咚咚，所以 ext2 规划出 inode 与 Block 来分别储存档案的属性(放在 inode 当中)与档案的内容(放置在 Block area 当中)。当我们要将一个 partition 格式化(format)为 ext2 时，就必须要指定 inode 与 Block 的大小才行，也就是说，当 partition 被格式化为 ext2 的档案系统时，他一定会有 inode table 与 block area 这两个区域。

Block 已经在前面说过了，他是数据储存的最小单位。那么 inode 是什么？！简单的说， Block 是记录『档案内容数据』的区域，至于 inode 则是记录『该档案的相关属性，以及档案内容放置在哪一个 Block 之内』的信息。 简单的说， inode 除了记录档案的属性外，同时还必须要具有指向(pointer)的功能，亦即指向档案内容放置的区块之中，好让操作系统可以正确的去取得档案的内容啊！底下几个是 inode 记录的信息(当然不止这些)：

- 该档案的拥有者与群组(owner/group)；
- 该档案的存取模式(read/write/excute)；
- 该档案的类型(type)；
- 该档案建立或状态改变的时间(ctime)、最近一次的读取时间(atime)、最近修改的时间(mtime)；
- 该档案的容量；
- 定义档案特性的旗标(flag)，如 SetUID...；
- 该档案真正内容的指向 (pointer)；

我们在前一章 Linux 档案与目录管理 当中提到过利用 ls 查询档案所记载的时间，就是 atime / ctime / mtime 三种时间。这三种时间的意义我们已经在前一章的 touch 指令介绍时提过，这三种时间就是记录在 inode 里面的啦～ 如果回到前一章，您会发现，我们可以利用 ls 的相关功能来查询到时间喔！而预设的显示时间是 mtime 。

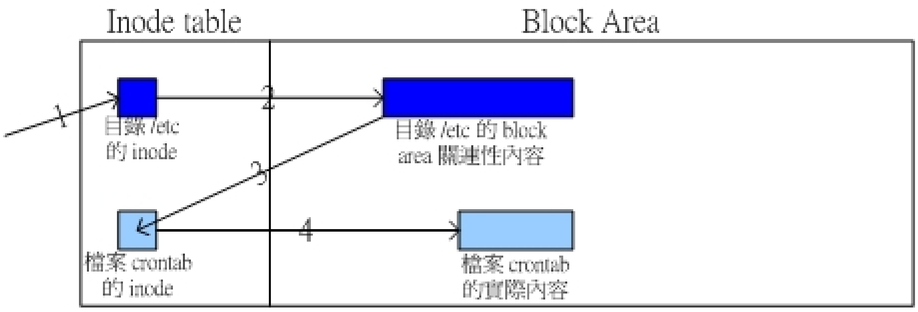
```
[root@linux ~]# ls -la --time=atime PATH
```

那个 PATH 是您所想要查询的档案或目录名称。利用上面的 ls 相关参数，就可以取得您想要知道的档案相关的三种时间啰～ 至于一个 inode 的大小为 128 bytes 这么大 (可以使用底下要介绍的 dumpe2fs 来查阅 inode 的大小喔！) ！好了，那么我的 Linux 系统到底是如何读取一个档案的内容呢？底下我们分别针对目录与档案来说明：

- 目录：
  当我们在 Linux 下的 ext2 档案系统建立一个目录时， ext2 会分配一个 inode 与至少一块 Block 给该目录。其中，inode 记录该目录的相关属性，并指向分配到的那块 Block ；而 Block 则是记录在这个目录下的相关连的档案(或目录)的关连性！

- 档案：
  当我们在 Linux 下的 ext2 建立一个一般档案时， ext2 会分配至少一个 inode 与相对于该档案大小的 Block 数量给该档案。例如：假设我的一个 Block 为 4 Kbytes ，而我要建立一个 100 KBytes 的档案，那么 linux 将分配一个 inode 与 25 个 Block 来储存该档案！

要注意的是， inode 本身并不纪录文件名，而是记录档案的相关属性，至于文件名则是记录在目录所属的 block 区域！ 那么档案与目录的关系又是如何呢？就如同上面的目录提到的，档案的相关连结会记录在目录的 block 数据区域， 所以当我们要读取一个档案的内容时，我们的 Linux 会先由根目录 / 取得该档案的上层目录所在 inode ， 再由该目录所记录的档案关连性 (在该目录所属的 block 区域) 取得该档案的 inode ， 最后在经由 inode 内提供的 block 指向，而取得最终的档案内容。我们以 /etc/crontab 这个档案的读取为例， 他的内容数据是这样取得的：



图三、读取 /etc/crontab 的简易流程示意。

一块 partition 在 ext2 底下会被格式化为 inode table 与 block area 两个区域， 所以在图三里面，我们将 partition 以长条的方式来示意，会比较容易理解的啦！而读取 /etc/crontab 的流程为：

1. 操作系统根据根目录(/)的相关资料可取得 /etc 这个目录所在的 inode ，并前往读取 /etc 这个目录的所有相关属性；
2. 根据 /etc 的 inode 的资料，可以取得 /etc 这个目录底下所有档案的关连数据是放置在哪一个 Block 当中，并前往该 block 读取档案的关连性内容；
3. 由上个步骤的 Block 当中，可以知道 crontab 这个档案的 inode 所在地，并前往该 inode ；
4. 由上个步骤的 inode 当中，可以取得 crontab 这个档案的所有属性，并且可前往由 inode 所指向的 Block 区域，顺利的取得 crontab 的档案内容。

整个读取的流程大致上就是这样，如果您想要实作一下以了解整个流程的话，可以这样试做看看：

```
1. 察看一下根目录所记载的所有档案关连性数据
[root@linux ~]# ls -lia /
      2 drwxr-xr-x  24 root root  4096 Jul 16 23:45 .
      2 drwxr-xr-x  24 root root  4096 Jul 16 23:45 ..
 719489 drwxr-xr-x  83 root root 12288 Jul 21 04:02 etc
 523265 drwxr-xr-x  24 root root  4096 Jun 25 20:16 var
# 注意看一下，在上面的 . 与 .. 都是连结到 inode 号码为 2 的那个 inode ，
# 也就是说， / 与其上层目录 .. 都是指向同一个 inode number 啊！两者是相同的。
# 而在根目录所记载的档案关连性 (在 block 内) 得到 /etc 的 inode number
# 为 719489 那个 inode number 喔！

2. 察看一下 /etc/ 内的档案关连性的数据
[root@linux ~]# ls -liad /etc/crontab /etc/.
719489 drwxr-xr-x  83 root root 12288 Jul 21 04:02 /etc/.
723496 -rw-r--r--   1 root root   663 Jul  4 12:03 /etc/crontab
```

```
# 瞧！此时就能够将 /etc/crontab 找到关连性啰！
```

所以您知道，目录的最大功能就是在提供档案的关连性，在关连性里面， 当然最主要的就是『档名与 inode 的对应数据』啰！另外，关于 EXT2 档案系统，这里有几点小事情要提醒一下：

- ext2 与 ext3 档案在建立时 (format) 就已经设定好固定的 inode 数与 block 数目了；
- 格式化 Linux 的 ext2 档案系统，可以使用 mke2fs 这个程序来执行！
- ext2 允许的 block size 为 1024, 2048 及 4096 bytes；
- 一个 partition (filesystem) 所能容许的最大档案数，与 inode 的数量有关， 因为一个档案至少要占用一个 inode 啊！
- 在目录底下的档案数如果太多而导致一个 Block 无法容纳的下所有的关连性数据时，Linux 会给予该目录多一个 Block 来继续记录关连数据；
- 通常 inode 数量的多寡设定为 (partition 的容量) 除以 (一个 inode 预计想要控制的容量)。举例来说，若我的 block 规划为 4Kbytes，假设我的一个 inode 会控制两个 block ，亦即是假设我的一个档案大致的容量在 8Kbytes 左右时，假设我的这个 partition 容量为 1GBytes， 则 inode 数量共有：(1G * 1024M/G * 1024K/M) / (8K) = 131072 个。而一个 inode 占用 128 bytes 的空间，因此格式化时就会有 (131072 个 * 128bytes/个) = 16777216 byes = 16384 Kbytes 的 inode table 。也就是说，这一个 1GB 的 partition 在还没有储存任何数据前， 就已经少了 16MBytes 的容量啊！
- 因为一个 inode 只能记录一个档案的属性，所以 inode 数量比 block 多是没有意义的！ 举上面的例子来说，我的 Block 规划为 4 Kbytes ，所以 1GB 大概就有 262144 个 4Kbytes 的 block ，如果一个 block 对应一个 inode 的话，那么当我的 inode 数量大于 262144 时，多的 inode 将没有任何用处，徒然浪费硬盘的空间而已！另外一层想法，如果我的档案容量都很大， 那么一个档案占用一个 inode 以及数个 block ，当然 inode 数量就可以规划的少很多啦！
- 当 block 大小越小，而 inode 数量越多，则可利用的空间越多，但是大档案写入的效率较差； 这种情况适合档案数量多，但是档案容量小的系统，例如 BBS 或者是新闻群组(News)这方面服务的系统；
- 当 Block 大小越大，而 inode 数量越少时，大档案写入的效率较佳，但是可能浪费的硬盘空间较多； 这种状况则比较适合档案容量较大的系统！

简单的归纳一下， ext2 有几个特色：

- Blocks 与 inodes 在一开始格式化时 (format) 就已经固定了；
- 一个 partition 能够容纳的档案数与 inode 有关；
- 一般来说，每 4Kbytes 的硬盘空间分配一个 inode ；
- 一个 inode 的大小为 128 bytes；
- Block 为固定大小，目前支持 1024/2048/4096 bytes 等；
- Block 越大，则损耗的硬盘空间也越多。

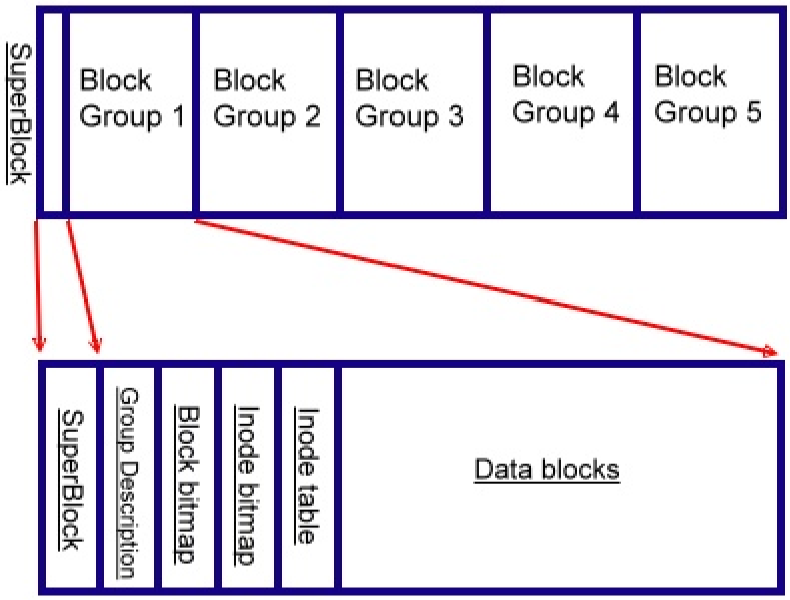
- 关于单一档案：
  若 block size=1024，最大容量为 16GB，若 block size=4096，容量最大为 2TB；
- 关于整个 partition ：
  若 block size=1024，则容量达 2TB，若 block size=4096，则容量达 32TB。
- 文件名最长达 255 字符，完整文件名长达 4096 字符。

另外，关于 partition 的使用效率上，当您的一个 partition 规划的很大时，例如 100GB 这么大， 由于硬盘上面的数据总是来来去去的，所以，整个 partition 上面的档案通常无法连续写在一起， 而是填入式的将数据填入没有被使用的 block 当中。如果档案写入的 block 真的分的很散， 此时就会有所谓的档案离散的问题发生了。虽然我们的 ext2 在 inode 处已经将该档案所记录的 block number 都记上了，所以资料可以一次性读取，但是如果档案真的太过离散，确实还是会发生读取效率低落的问题。 果真如此，那么可以将整个 partition 内的数据全部复制出来，将该 partition 重新格式化， 再将数据给他复制回去即可解决。

此外，如果 partition 真的太大了，那么当一个档案分别记录在这个 partition 的最前面与最后面的 block， 此时会造成硬盘的机械手臂移动幅度过大，也会造成数据读取效能的低落。因此， partition 的规划并不是越大越好， 而是真的要针对您的主机用途来进行规划才行！ ^_^

EXT2/EXT3 档案的存取与日志式档案系统的功能

综合上面谈的种种，我们可以知道，当一个 ext2 的 filesystem 被建立时， 他拥有 superblock / group description / block bitmap / inode bitmap / inode table / data blocks 等等区域。要注意的是，每个 ext2 filesystem 在被建立的时候，会依据 partition 的大小， 给予数个 block group ，而每个 block group 就有上述的这些部分。整个 filesystem 的架构可以下图展现：



图四、整个 filesystem 的展现示意图

我们将整个 filesystem 简单化， 假设仅有一个 block group ，那么上面的各个部分分别代表什么呢？

- SuperBlock：如前所述， Superblock 是记录整个 filesystem 相关信息的地方， 没有 Superblock ，就没有这个 filesystem 了。他记录的信息主要有：
    - block 与 inode 的总量；
    - 未使用与已使用的 inode / block 数量；
    - 一个 block 与一个 inode 的大小；
    - filesystem 的挂载时间、最近一次写入数据的时间、最近一次检验磁盘 (fsck) 的时间等档案系统的相关信息；
    - 一个 valid bit 数值，若此档案系统已被挂载，则 valid bit 为 0 ，若未被挂载，则 valid bit 为 1 。
- Group Description：纪录此 block 由由何处开始记录；
- Block bitmap：此处记录那个 block 有没有被使用；
- Inode bitmap：此处记录那个 inode 有没有被使用；
- Inode table：为每个 inode 数据存放区；
- Data Blocks：为每个 block 数据存放区。

如果想要知道某个 ext2/ext3 的档案系统内，关于上述提到的相关信息时，可以使用 dumpe2fs 这个指令来读取，举例来说，鸟哥将我自己的主机 /dev/hda1 读出 ext3 的讯息：

```
[root@linux ~]# dumpe2fs /dev/hda1
Filesystem volume name:   /
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              1537088
Block count:              1536207
Free blocks:              735609
Free inodes:              1393089
First block:              0
Block size:               4096
Filesystem created:       Sat Jun 25 16:21:13 2005
Last mount time:          Sat Jul 16 23:45:04 2005
Last write time:          Sat Jul 16 23:45:04 2005
Last checked:             Sat Jun 25 16:21:13 2005
First inode:              11
Inode size:               128
Journal inode:            8

Group 0: (Blocks 0-32767)
  Primary superblock at 0, Group descriptors at 1-1
  Reserved GDT blocks at 2-376
  Block bitmap at 377 (+377), Inode bitmap at 378 (+378)
  Inode table at 379-1400 (+379)
  0 free blocks, 32424 free inodes, 11 directories
  Free blocks:
```

```
  Free inodes: 281-32704
Group 1: (Blocks 32768-65535)
  Backup superblock at 32768, Group descriptors at 32769-32769
  Reserved GDT blocks at 32770-33144
  Block bitmap at 33145 (+377), Inode bitmap at 33146 (+378)
  Inode table at 33147-34168 (+379)
  18 free blocks, 24394 free inodes, 349 directories
  Free blocks: 37882-37886, 38263-38275
  Free inodes: 38084-38147, 39283-39343, 41135, 41141-65408
# 因为数据很多，所以鸟哥略去了一些信息了～上面是比较精简的显示内容。
# 在 Group 0 之前的都是 Superblock 的内容，记录了 inode/block 的总数，
# 还有其它相关的讯息。至于由 Group 0 之后，则是说明各个 bitmap 及 inode table
# 与 block area 等等。
```

透过这些记录，我们可以很轻易的就知道哪些 inode 没有被使用，哪些 block 还可以记录， 如此一来，在新增、建立档案与目录时，系统就会根据这些记录来将数据分别写入尚未被使用的 inode 与 block area 了！ 不过，要注意的是，当我们新增一个档案(目录)时：

1. 根据 inode bitmap / block bitmap 的信息，找到尚未被使用的 inode 与 block ， 进而将档案的属性与数据分别记载进 inode 与 block ；
2. 将刚刚被利用的 inode 与 block 的号码 (number) 告知 superblock、inode bitmap、block bitmap 等，让这些 metadata 更新信息。

一般来说，我们将 inode table 与 block area 称为数据存放区域，至于其它的例如 superblock、 block bitmap 与 inode bitmap 等记录就被称为 metadata 啰。经由上面两个动作，我们知道一笔数据写入硬盘时， 会有这两个动作。

- 数据的不一致 (Inconsistent) 状态

那么万一您的档案在写入硬盘时，因为不知名原因导致系统中断(例如突然的停电啊、 系统核心发生错误啊～等等的怪事发生时)，所以数据就只有纪录到动作一，而动作二尚未进行～ 这就会产生 metadata 与数据存放区产生不一致 (Inconsistent) 的情况发生了。

在早期的 EXT2 档案系统中，如果发生这个问题，那么系统在重新开机的时候， 就会藉由 Superblock 当中记录的 valid bit 与 filesystem state 等状态来判断是否强制进行数据一致性的检查！检查则以 e2fsck 这支程序来进行的。 不过，这样的检查真的是很费时～因为要针对 metadata 区域与实际数据存放区来进行比对， 呵呵～得要搜寻整个 partition 呢～哇！系统真忙碌～而且在对 Internet 提供服务的服务器主机上面， 这样的检查真的会造成主机复原时间的拉长～真是麻烦～这也就造成后来所谓日志式档案系统的兴起了。

稍微了解了所谓数据不一致的状态后，再来要了解的，就是，那么为何要有日志式档案系统的产生呢？ 我们已经在 Linux 档案属性与目录配置 当中提到过一些档案系统的注意事项， 也提过日志式 (Journal) 档案系统的相关功能，这里我们再稍微深入的讨论一下。

刚刚提到了，在 EXT2 档案系统当中，要进行档案的写入时，会将数据分别在数据存放区与 metadata 区记录下来， 若当这两个动作无法一次完成时，就会造成所谓的不一致现象。若发生不一致现象， 因为系

统不知道是那个档案发生不一致现象，所以就会将整个 filesystem 做一致性的检查，如此一来，很费时啊！ 想一想，如果在我们的 filesystem 当中，要是能够规划出一个区块，专门来记录写入或修订档案时的步骤， 那不就可以简化一致性检查的步骤了？也就是说：

1. 当系统要写入一个档案的时候，会先在日志记录区块中纪录：某个档案准备要写入磁盘了；
2. 开始写入档案的权限与数据；
3. 开始更新 metadata 的数据；
4. 完成数据与 metadata 的更新后，在日志记录区块当中完成该档案的纪录。

在这样的程序当中，万一数据的纪录过程当中发生了问题，那么我们的系统只要去检查日志记录区块， 就可以知道那个档案发生了问题，针对该问题来做一致性的检查即可，而不必针对整块 filesystem 去检查，真的就可以达到快速修复 filesystem 的能力了！这就是日志式档案最基础的功能啰～ 那么我们的 ext2 可达到这样的功能吗？当然可以啊！ 就透过 ext3 即可！ ext3 是 ext2 的升级版本，并且可向下兼容 ext2 版本呢！ 所以啰，目前我们才建议大家，可以直接使用 ext3 这个 filesystem 啊！ ^_^

如果您对于 EXT2 / EXT3 系统还有更多的兴趣，可以参考底下这几篇文章：

- Design and Implementation of the Second Extended Filesystem
  http://e2fsprogs.sourceforge.net/ext2intro.html
- The Second Extended File System - An introduction
  http://www.freeos.com/articles/3912/
- ext3 or ReiserFS? Hans Reiser Says Red Hat's Move Is Understandable
  http://www.linuxplanet.com/linuxplanet/reports/3726/1/

或者参考鸟哥由网络上面找到的相关中文翻译，不过.....原发文者的文章出处已经找不到了～ 真是不好意思～请参考：附录 B：EXT2/EXT3 档案系统。

Linux 档案系统的运作：

好了，我们知道整个 ext2/ext3 的数据存取是透过 journal 与 metadata 还有数据存放区在纪录的。 不过，实际上， Linux 档案系统在运作的时候，真的要将数据直接存放到硬盘上面吗？！ 有没有更有效率的作法？

我们来看看整部计算机的运作当中，那个数据的存取速度最慢呢？数据处理最快速的地方应该是 CPU 了，接下来则是主存储器 (RAM) ，至于硬盘，哈哈！没错，速度可是比 CPU 还有 RAM 要慢的很多很多。 为了让 Linux 加快整个系统的存取效率，因此在 Linux 上面通常采取异步处理(asynchronously)的方式。

什么是异步呢？举例来说：『当系统读取了某一个档案， 则该档案所在的区块数据会被加载到内存当中，所以该磁盘区块就会被放置在主存储器的缓冲快取区中， 若这些区块的数据被改变时，刚开始数据仅有主存储器的区块数据会被改变， 而且在缓冲区当中的区块数据会被标记为『 Dirty 』，这个时候磁盘实体区块尚未被修正！ 所以亦即表示，这些『 Dirty 』区块的数据必需回写到磁盘当中， 以维持磁盘实体区块上的数据与主存储器中的区块数据的一致性。』

为什么要这么做呢？这是因为主存储器的运作速度比起硬盘来实在是快太多了，万一系统当中有一个档案

相当的大，而又持续性的存取，那么由于较慢的硬盘存取速度，将使得整个 Linux 速度被拖垮，所以才会使用异步方式的数据处理啊！不过， 也由于硬盘与主存储器的数据可能没有同步化，因此，如果 Linux 不正常关机(例如跳电或者是当机)时，则由于数据尚未回写入磁盘当中，会导致系统在再次开机时，会花相当多的时间进行磁盘检验， 同时也有可能造成磁盘的损毁啊！

挂载点的意义 (mount point)：

我们上面提到的都是关于档案系统 (filesystem)，但是要能够让我们的 Linux 使用的话， 非得『挂载 (mount)』上我们的 Linux 系统才行啊！ 刚刚我们上面提到了目录可以记录文件名与 inode 的相关信息，此外， 目录也是让我们得以跟 filesystem 产生对应的入口点。因此，我们称那个入口点目录为『 挂载点 (mount point) 』

举例来说，在鸟哥的 安装 FC4 范例当中，我们将硬盘分割为几大部分， 同时主要将 / 与 /home 设定为两个 partition 的挂载点。假设 / 是接在 /dev/hda1 ， 而 /home 是接在 /dev/hda2 上面，那么，也就是说，在 /home 底下的所有次目录，使用的都是 /dev/hda2 那个 partition 的资料呢！而非 /home 的则都是使用 /dev/hda1 的数据！

那么来看看系统中如果主要分为 / 与 /home 时，他们对应的 inode 会有什么现象呢？

```
[root@linux ~]# ls -lid / /home
2 drwxr-xr-x  26 root root 4096  7 月 21 09:08 /
2 drwxr-xr-x  42 root root 4096  7 月 14 23:37 /home
```

看到了吧？咦！怎么 / 与 /home 的 inode number 都是 2 啊？？这太不合理了～ 原因很简单啊！因为 / 是 /dev/hda1 而 /home 是 /dev/hda2 ，这两个 partition 都有 inode number 为 2 的号码啊！所以啊，请注意， 挂载点一定是『目录』而不是档案喔！ 也就是说，这个挂载点就是进入该 filesystem 的入口啦！

其它 Linux 支持的档案系统

虽然 Linux 的标准档案系统是 ext2 ，且还有增加了日志功能的 ext3 之外，事实上，Linux 还有支持很多档案格式的，尤其是最近这几年推出了好几种速度很快的日志式档案系统，包括 SGI 的 XFS 档案系统，可以适用更小型档案的 Reiserfs 档案系统，以及 Windows 的 FAT 档案系统等等， 都能够被 Linux 所支持喔！常见的支持档案系统有：

- 传统档案系统：ext2 / minix / MS-DOS / FAT (用 vfat 模块) / iso9660 (光盘)等等；
- 日志式档案系统： ext3 / ReiserFS / Windows' NTFS / IBM's JFS / SGI's XFS
- 网络档案系统： NFS / SMBFS

想要知道您的 Linux 支持的档案系统有哪些，可以察看底下这个目录：

```
[root@linux ~]# ls -l /lib/modules/`uname -r`/kernel/fs
```

系统目前已启用的档案系统则有：

```
[root@linux ~]# cat /proc/filesystems
```

假设您的 / 使用的是 /dev/hda1 ，用 ext3 ，而 /home 使用 /dev/hda2 ，用 reiserfs ， 那么您取用 /home/dmtsai/.bashrc 时，有特别指定要用的什么档案系统的模块来读取吗？！ 应该是没有吧！嘿嘿！这个就是我们 Linux kernel 的 Virtual Filesystem Switch (VFS) 的功能啦！透过这个 VFS 的功能来管理所有的 filesystem， 省去我们需要自行设定读取档案系统的定义啊～方便很多！

档案系统的简单操作：

在了解了一些简单的硬盘与档案系统的概念之后，并且知道如何以 ls 查询档案系统相关的信息后， 接下来就是得要了解如何知道整个磁盘的剩余容量与总容量啰～此外，也得要知道一下，前一章还没有介绍到的连结档 (link file) 啰～

磁盘与目录的容量：

在文字接口底下有什么方法可以查看目前的磁盘最大容许容量、已经使用掉的容量、 目前所在目录的已使用容量？还有还有，怎么知道目前目录底下使用掉的硬盘容量呢？ 以及如何查询目前的 inodes 数目？呵呵！底下我们就来谈一谈主要的两个指令：

- df

```
[root@linux ~]# df [-ahikHTm] [目录或文件名]
参数：
-a  ：列出所有的档案系统，包括系统特有的 /proc 等档案系统；
-k  ：以 KBytes 的容量显示各档案系统；
-m  ：以 MBytes 的容量显示各档案系统；
-h  ：以人们较易阅读的 GBytes, MBytes, KBytes 等格式自行显示；
-H  ：以 M=1000K 取代 M=1024K 的进位方式；
-T  ：连同该 partition 的 filesystem 名称 (例如 ext3) 也列出；
-i  ：不用硬盘容量，而以 inode 的数量来显示
范例：
范例一：将系统内所有的 partition 列出来！
[root@linux ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1              5952252   3012332   2632680  54% /
/dev/shm                192836         0    192836   0% /dev/shm
/dev/hda5              9492644    221604   8781060   3% /home
# 特别注意，在 Linux (FC4) 底下，如果 df 没有加任何参数，
# 那么预设会将系统内所有的 (不含特殊内存内的档案系统与 swap) 都以 Kbytes
# 的容量来列出来！至于那个 /dev/shm 是与内存有关的挂载，先不要理他！

范例二：将容量结果以易读的容量格式显示出来
[root@linux ~]# df -h
```

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             5.7G  2.9G  2.6G  54% /
/dev/shm              189M     0  189M   0% /dev/shm
/dev/hda5             9.1G  217M  8.4G   3% /home
# 不同于范例一，这里会以 G/M 等容量格式显示出来，比较容易看啦!

范例三：将系统内的所有特殊档案格式及名称都列出来
[root@linux ~]# df -aT
Filesystem    Type   1K-blocks      Used Available Use% Mounted on
/dev/hda1     ext3     5952252   3012332   2632680  54% /
/dev/proc     proc           0         0         0   -  /proc
/dev/sys     sysfs           0         0         0   -  /sys
/dev/devpts devpts           0         0         0   -  /dev/pts
/dev/shm     tmpfs      192836         0    192836   0% /dev/shm
/dev/hda5     ext3     9492644    221604   8781060   3% /home
none   binfmt_misc           0         0         0   -  /proc/sys/fs/binfmt_misc
# 看到了吧！系统里面其实还有很多的特殊档案系统在跑得！
# 不过，那些比较特殊的档案系统几乎都是在内存当中，例如 /proc 这个挂载点。
# 因此，这些特殊的档案系统都不会占据硬盘空间喔！ ^_^

范例四：将 /etc 底下的可用的磁盘容量以易读的容量格式显示
[root@linux ~]# df -h /etc
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             5.7G  2.9G  2.6G  54% /
# 这个范例比较有趣一点啦，在 df 后面加上目录或者是档案时， df
# 会自动的分析该目录或档案所在的 partition ，并将该 partition 的容量显示出来，
# 所以，您就可以知道某个目录底下还有多少容量可以使用了！ ^_^

范例五：将目前各个 partition 当中可用的 inode 数量列出
[root@linux ~]# df -ih
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda1               1.5M    141K    1.4M   10% /
/dev/shm                 48K       1     48K    1% /dev/shm
/dev/hda5               2.4M      67    2.4M    1% /home
```

这是用来显示目前磁盘的总容量与剩余可用容量的指令！需要注意的是， 由于我们的档案或者是外挂的磁盘都是加在『 / 』底下，所以当根目录没有空间的时候， 嘿嘿！你的 Linux 系统可能大概就要挂了吧～当然啰！你可以将你的资料放置在加挂的硬盘中， 那么如何知道目前哪一个磁盘还有多少空间呢？！

Tips:
说个笑话！当初我们系上有个研究生在管理 Sun 的工作站，是别研究室的， 他的硬盘明明有好几 GB ，但是就是没有办法将几 MB 的数据 copy 进去， 他就去跟老板讲说机器坏了！嘿！明明才来维护过几天而已为何会坏了！ 结果老板将维护商叫来骂了 2 小时左右吧！后来，维护商发现原来硬盘的『总空间』还有很多， 只



是某个扇区填满了，偏偏该研究生就是要将数据 copy 去那个扇区！呵呵！ 后来那个研究生就被命令『再也不许碰 Sun 主机』了～～

这里要请大家再复习一下，我们的硬盘扇区规划中， primary 扇区每一颗硬盘最多只允许 4 个，其它的就放置在 Extended 扇区中了！而，硬盘的代号与 IDE 的插槽是有关系的！ 如果忘记了，那就回去安装 Linux 那一章复习一下吧！好了！假设我只有一棵硬盘，且放在 IDE 的 master ，那么我的硬盘就是 /dev/hda 啰！而在这颗硬盘中的分割扇区就可以由 /dev/hda1 开始向上加！

OK，那么使用 df -k 之后，假设我的硬盘分为 /dev/hda1, /dev/hda2, /dev/hda3, /dev/hda5 与 /dev/hda6 ，咦！ /dev/hda4 跑去哪里了！呵呵！其实 /dev/hda4 通常就是 Extended 扇区啦！而后面的 /dev/hda5, /dev/hda6 等扇区就是由 /dev/hda4 所切出来的！所以 /dev/hda5 + /dev/hda6 = /dev/hda4 ！当然，如果还有没有显示出来的，例如 Swap ，则 /dev/hda4 还有可能更大呦！

那么来解释一下上面的数据吧！

- Filesystem：代表该档案系统是在那个 partition 啊，所以列出装置名称;
- 1k-blocks：说明底下的数字单位是 1KB 呦！可利用 -h 或 -m 来改变容量;
- Used：顾名思义，就是使用掉的硬盘空间啦！
- Available：也就是剩下的磁盘空间大小;
- Use%：就是磁盘的使用率啦！如果使用率高达 90% 以上时， 最好需要注意一下了，免得容量不足造成系统问题喔！ (例如最容易被灌爆的 /var/spool/mail 这个放置邮件的磁盘)
- Mounted on：就是磁盘挂载的目录所在啦！(挂载点啦！)

另外，需要注意的是，如果使用 -a 这个参数时，系统会出现 /proc 这个扇区，但是里面的东西都是 0 ，不要紧张！ /proc 的东西都是 Linux 系统所需要加载的系统数据，而且是挂载在『内存当中』的， 所以当然没有占任何的硬盘空间啰！

- du

```
[root@linux ~]# du [-ahskm] 档案或目录名称
参数:
-a  : 列出所有的档案与目录容量，因为预设仅统计目录底下的档案量而已。
-h  : 以人们较易读的容量格式 (G/M) 显示;
-s  : 列出总量而已，而不列出每个各别的目录占用容量;
-k  : 以 KBytes 列出容量显示;
-m  : 以 MBytes 列出容量显示;
范例:
范例一：列出目前目录下的所有档案容量
[root@linux ~]# du
[root@vbird ~]# du
16      ./.gnome2
16      ./.ssh
.......中间省略.......
292     .
```

```
# 直接输入 du 没有加任何参数时，则 du 会分析『目前所在目录』
# 的档案与目录所占用的硬盘空间。但是，实际显示时，仅会显示目录容量，
# 但我的 . 目录有很多档案没有被列出来，所以，全部的目录相加不会等于 . 的容量喔！

范例二：同范例一，但是将档案的容量也列出来
[root@linux ~]# du -a
12      ./install.log.syslog
16      ./.gnome2
16      ./.ssh
76      ./install.log
16      ./.bash_history
4       ./.bashrc
.......中间省略........
292     .
# 加上这个 -a 参数后，就会将目录底下的档案也一起列示出来，
# 而不是仅列出档案信息而已！注意啰～ ^_^

范例三：检查根目录底下每个目录所占用的容量
[root@linux ~]# du -sm /*
7       /bin
14      /boot
.......中间省略........
385     /proc
.......中间省略........
1       /tmp
2944    /usr
79      /var
# 这是个很常被使用的功能啰～利用万用字符 * 来代表每个目录，
# 所以，如果想要检查某个目录下，那个次目录占用最大的容量，就可以用这个方法找出来
# 值得注意的是，如果您刚刚安装好 Linux 时，那么整个系统容量最大的应该是 /usr
# 那个目录，而 /proc 虽然有列出容量，但是那个容量是在内存中，不占硬盘空间。
```

在 Windows 底下可以使用档案总管来管理你的磁盘，在 Linux 底下也可以轻易的以 du 来知道目前磁盘的档案容量耶！在预设的情况下，容量的输出是以 KB 来设计的， 如果你想要知道目录占了多少 MB ，那么就使用 -m 这个参数即可啰！而， 如果你只想要知道该目录占了多少容量的话，呵呵，使用 -s 就可以啦！另外， 如同上面的范例三，可以利用万用字符 * 来加快你的搜寻喔！

连结档的介绍： ln

什么是连结档呢？其实连结档有点类似 Windows 底下的『快捷方式』！也就是很多的连结档案(link file)其实都指向同一个来源档案(source file)！不过，在所有的档案类型当中， 连结档算是比较难理解的一部份了！因为连结档还分成 Hard link 与 symbolic link 两种，这两种连结档在架构上是完全不一样的咚咚，底下就来好好的谈一谈先！

- Hard Link (硬式连结或实际连结)

在前一节当中，我们提到档案的读取方式为：

1. 先由一层一层的目录取得档案相关的关连数据，
2. 再到对应的 inode 取得档案的属性，以及档案内容数据所在的 Block ，
3. 最后到 Block area 取得档案的数据。

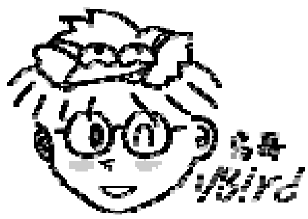
那么 hard link 怎么制作档案的连结呢？！很简单， Hard Link 只是在某个目录下新增一个该档案的关连数据而已！

举个例子来说，假设我的 /root/crontab 为一个 hard link 的档案，他连结到 /etc/crontab 这个档案，也就是说，其实 /root/crontab 与 /etc/crontab 是同一个档案，只是有两个目录(/etc 与 /root)记录了 crontab 这个档案的关连数据罢了！也就是说，我由 /etc 这个目录所记录的关连数据可知道 crontab 的 inode 放置在 A 处，而由 /root 这个目录下的关连数据， crontab 同样也指到 A 处的 inode ！所以啰， crontab 这个档案的 inode 与 block 都没有改变， 有的只是有两个目录记录了关连数据。

那这样有什么好处呢？最大的好处就是『安全！』如同上面提到的 /root/crontab 与 /etc/crontab 中，不管哪一个档案被删除了,其实仅是移除一笔目录底下的档案关连性数据,并没有更动到原本档案的 inode 与 block 数据呢!而且,不论由那个目录连结到正确的 crontab 的 inode 与 block ， 都可以正确无误的进行数据的修改喔！ ^_^

一般来说，使用 hard link 设定连结文件时，磁盘的空间与 inode 的数目都不会改变！ 由上面的说明来看，我们可以知道， hard link 只是在某个目录下的 block 多写入一个关连数据，所以当然不会用掉 inode 与磁盘空间啰！

> Tips:
> 其实可能会改变的，那就是当目录的 Block 被用完时，就可能会新加一个 block 来记录,而导致磁盘空间的变化!不过,一般 hard link 所用掉的关连数据量很小,所以通常不会改变 inode 与磁盘空间的大小喔！



由于 hard link 是在同一个 partition 上面进行数据关连的建立，所以 hard link 是有限制的：

- 不能跨 Filesystem;
- 不能 link 目录。

不能跨 Filesystem 还好理解，因为 hard link 本来就是在一个 partition 内建立关连性的， 那不能 hard link 到目录又是怎么回事呢？这是因为如果使用 hard link 连结到目录时， 连结的数据被需要连同被连结目录底下的所有数据都建立连结，举例来说，如果你要将 /etc 使用硬式连结建立一个 /etc_hd 的目录时，那么在 /etc_hd 底下的所有数据同时都与 /etc 底下的数据要建立 hard link 的，而不能仅是连结到 /etc_hd 与 /etc 而已。并且,未来如果需要在 /etc_hd 底下建立新档案时,连带的，/etc 底下的数据又得要建立一次 hard link ，因此造成环境相当大的复杂度。 所以啰，目前 hard link 对于目录暂时还是不支持的啊！

- Symbolic Link (符号连结，亦即是快捷方式)

相对于 hard link ， Symbolic link 可就好理解多了，基本上， Symbolic link 就是在建立一个独立的档案， 而这个档案会让数据的读取指向他 link 的那个档案内容！由于只是利用档案来做为指向的动作，所以，当来源档被删除之后，symbolic link 的档案会『开不了』， 会一直说『无法开启某档案！』。这里还是得特别留意，这个 Symbolic Link 与 Windows 的快捷方式可以给他划上等号，由 Symbolic link 所建立的档案为一个独立的新的档案，所以会占用掉 inode 与 block 喔！

由上面的说明来看，似乎 hard link 比较安全，因为即使某一个目录下的关连数据被杀掉了， 也没有关系，只要有任何一个目录下存在着关连数据，那么该档案就不会不见！举上面的例子来说，我的 /etc/crontab 与 /root/crontab 指向同一个档案，如果我删除了 /etc/crontab 这个档案，该删除的动作其实只是将 /etc 目录下关于 crontab 的关连数据拿掉而已， crontab 所在的 inode 与 block 其实都没有被变动喔！

不过，不幸的是，由于 Hard Link 的限制太多了，包括无法做『目录』的 link ， 所以在用途上面是比较受限的！反而是 Symbolic Link 的使用方面较广喔！好了， 说的天花乱坠，看您也差不多快要昏倒了！没关系，实作一下就知道怎么回事了！要制作连结档就必须要使用 ln 这个指令呢！

```
[root@linux ~]# ln [-sf] 来源文件 目标文件
参数：
-s  ：如果 ln 不加任何参数就进行连结，那就是 hard link，至于 -s 就是 symbolic link
-f  ：如果 目标文件 存在时，就主动的将目标文件直接移除后再建立！
范例：
范例一：将 /etc/passwd 复制到 /tmp 底下，并且观察 inode 与 block
[root@linux ~]# cd /tmp
[root@linux tmp]# cp -a /etc/passwd .
[root@linux tmp]# du -sb ; df -i .
26948   . <== 先注意一下，这里的容量是多少！
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda1            1537088  144016 1393072   10% /
# 利用 du 与 df 来检查一下目前的参数～那个 du -sb
# 是计算整个 /tmp 底下有多少 bytes 的容量啦！

范例二：将 /tmp/passwd 制作 hard link 成为 passwd-hd 档案
[root@linux tmp]# ln passwd passwd-hd
[root@linux tmp]# du -sb ; df -i .
26948   .
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda1            1537088  144016 1393072   10% /
# 仔细看，即使多了一个档案在 /tmp 底下，整个 inode 与 block 的容量并没有改变！
[root@linux tmp]# ls -il passwd*
1242760 -rw-r--r--  2 root root 1746 Jun 29 01:03 passwd
1242760 -rw-r--r--  2 root root 1746 Jun 29 01:03 passwd-hd
# 原来是指向同一个 inode 啊！这是个重点啊！另外，那个第二栏的连结数也会增加！

范例三：将 /tmp/passwd 建立一个符号连结
[root@linux tmp]# ln -s passwd passwd-so
```

```
[root@linux tmp]# ls -li passwd*
1242760 -rw-r--r--  2 root root 1746 Jun 29 01:03 passwd
1242760 -rw-r--r--  2 root root 1746 Jun 29 01:03 passwd-hd
1242806 lrwxrwxrwx  1 root root    6 Jul 23 20:02 passwd-so -> passwd
# 仔细看喔，这个 passwd-so 指向的 inode number 不同了！这是一个新的档案～
# 这个档案的内容是指向 passwd 的，你可以看到这个档案的大小，是 6bytes ，
# 怎么来的？因为 passwd 共有六个字符啊！哈哈！没错～这个连结档的内容只是填写
# 连结的目标档案文件名而已！所以，你的连结档档名（有时候含路径）有多长，档案就多大！
[root@linux tmp]# du -sb ; df -i .
26954   .
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda1            1537088  144017 1393071   10% /
# 呼呼！整个容量与 inode 使用数都改变啰～确实如此啊！

范例四：删除源文件 passwd ，其它两个档案是否能够开启？
[root@linux tmp]# rm passwd
[root@linux tmp]# cat passwd-hd
......正常显示完毕！
[root@linux tmp]# cat passwd-so
cat: passwd-so: No such file or directory
# 怕了吧？！竟然无法正常的开启这个档案呢～
```

Tips:
还记得上一章当中，我们提到的 /tmp 这个目录是干嘛用的吗？是给大家作为暂存盘用的啊！ 所以，您会发现，过去我们在进行测试时，都会将数据移动到 /tmp 底下去练习～ 嘿嘿！因此，有事没事，记得将 /tmp 底下的一些怪异的数据清一清先！



要注意啰！使用 ln 如果不加任何参数的话，那么就是 Hard Link 啰！如同上面的情况，增加了 hard link 之后，可以发现使用 ls -l 时，显示的 link 那一栏属性增加了！而如果这个时候砍掉 passwd 会发生什么事情呢？呵呵！ passwd-hd 的内容还是会跟原来 passwd 相同，但是 passwd-so 就会找不到该档案啦！就是这样！了解了吗？！

而如果 ln 使用 -s 的参数时，就做成差不多是 Windows 底下的『快捷方式』的意思(Symbolic Link，较常用！)。当你修改 Linux 下的 link 档案时，则更动的其实是『原始档』，呵呵， 所以不论你的这个原始档被连结到哪里去，只要你修改了连结档，呵呵！原始档就跟着变啰！ 以上面为例，由于你使用 -s 的参数建立一个名为 passwd-so 的档案，则你修改 passwd-so 时，其内容与 passwd 完全相同，并且，当你按下储存之后，被改变的将是 passwd 这个档案！

此外，如果你做了底下这样的连结：

```
ln -s /bin /root/bin
```

那么如果你进入 /root/bin 这个目录下，『请注意呦！该目录其实是 /bin 这个目录，因为你做了连结档了！』所以，如果你进入 /root/bin 这个刚刚建立的连结目录， 并且将其中的数据杀掉时，嗯！ /bin 里面的数据就通通不见了！这点请千万注意！并不是 /root 底下的资料都是 root 的！还需要注意一下该属

性才行！(其实可以透过 pwd -P 去观察！)

基本上， Symbolic link 的用途比较广，所以您要特别留意 symbolic link 的用法呢！未来一定还会常常用到的啦！

- 关于目录的 link 数量：

或许您已经发现了，那就是，当我们以 hard link 进行『档案的连结』时，可以发现，在 ls -l 所显示的第二字段会增加一才对，那么请教，如果建立目录时，他预设的 link 数量会是多少？ 让我们来想一想，一个『空目录』里面至少会存在些什么？呵呵！就是存在 . 与 .. 这两个目录啊！ 那么，当我们建立一个新目录名称为 /tmp/testing 时，基本上会有三个东西，那就是：

- /tmp/testing
- /tmp/testing/.
- /tmp/testing/..

而其中 /tmp/testing 与 /tmp/testing/. 其实是一样的！都代表该目录啊～而 /tmp/testing/.. 则代表 /tmp 这个目录，所以说，当我们建立一个新的目录时，『新的目录的 link 数为 2 ，而上层目录的 link 数则会增加 1 』 不信的话，我们来作个测试看看：

```
[root@linux ~]# ls -ld /tmp
drwxrwxrwt  5 root root 4096 Oct 11 05:15 /tmp
[root@linux ~]# mkdir /tmp/testing1
[root@linux ~]# ls -ld /tmp
drwxrwxrwt  6 root root 4096 Oct 11 13:58 /tmp
[root@linux ~]# ls -ld /tmp/testing1
drwxr-xr-x  2 root root 4096 Oct 11 13:58 /tmp/testing1
```

瞧！原本的所谓上层目录 /tmp 的 link 数量由 5 增加为 6 ，至于新目录 /tmp/testing 则为 2 ，这样可以理解目录的 link 数量的意义了吗？！ ^_^

磁盘的分割、格式化、检验与挂载：

对于一个系统管理者 (root) 而言，磁盘的的管理是相当重要的一环，尤其近来硬盘已经渐渐的被当成是消耗品了 ..... 好了，如果我们想要在系统里面新增一颗硬盘时，应该有哪些动作需要做的呢？有几个动作啰：

1. 对磁盘进行分割，以建立可用的 partition ；
2. 对该 partition 进行格式化(format)，以建立系统可用的 filesystem；
3. 若想要仔细一点，则可对刚刚建立好的 filesystem 进行检验；
4. 在 Linux 系统上，需要建立挂载点 (亦即是目录)，并将他挂载上来；

当然啰，在上述的过程当中，还有很多需要考虑的，例如磁盘分割槽 (partition) 需要定多大？ 是否需要加入 journal 的功能？inode 与 block 的数量应该如何规划等等的问题。但是这些问题的决定， 都需要与您的主机用途来加以考虑的～所以，在这个小节里面，鸟哥仅会介绍几个动作而已， 更详细的设定值，则需要以您未来的经验来参考啰！

磁盘分割： fdisk

```
[root@linux ~]# fdisk [-l] 装置名称
参数：
-l  ：输出后面接的装置所有的 partition 内容。若仅有 fdisk -l 时，
      则系统将会把整个系统内能够搜寻到的装置的 partition 均列出来。
范例：
范例：查阅您的第一颗硬盘内的相关信息
[root@linux ~]# fdisk /dev/hda  <== 仔细看，不要加上数字喔！
The number of cylinders for this disk is set to 2494.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
# 在你进入 fdisk 这支程序的工作画面后，如果您的硬盘太大的话，就会出现如上讯息。
# 这个讯息仅是在告知你，因为某些旧版的软件与操作系统并无法支持大于 1024
# 磁柱 (cylinter) 后的扇区使用，不过我们 Linux 是没问题啦！

Command (m for help): m   <== 输入 m 后，就会看到底下这些指令介绍
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
# 这里注意一下，使用 fdisk 这支程序是完全不需要背指令的，因为按下 m 之后，
# 立刻就会有一堆指令说明跑出来了！在上面的指令当中，比较重要的有：
# d 删除一个磁盘分割槽、 n 新增一个磁盘分割槽、 p 将目前的磁盘分割槽列出来、
# q 不储存离开！这个重要！ w 写入磁盘分割表后离开！这个危险！
```

```
Command (m for help): p  <== 这里可以输出目前磁盘的状态

Disk /dev/hda: 20.5 GB, 20520493056 bytes <== 硬盘的信息在这底下三行
255 heads, 63 sectors/track, 2494 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1         765     6144831   83  Linux
/dev/hda2             766        1147     3068415   83  Linux
/dev/hda3            1148        1274     1020127+  82  Linux swap
/dev/hda4            1275        2494     9799650    5  Extended
/dev/hda5            1275        2494     9799618+  83  Linux
# 由上面的信息，您可以知道，我的硬盘是 20GB 的，而 Head/Sector/Cylinder
# 的数量为 255/63/2494 ，另外，可以看到上头的 Boot 吗？那个地方代表有开机信息的
# partition ！另外，那个 start 与 end 则是指每一个 partition 的开始与结束的
# Cylinder 号码！这样可以了解我们前面一直强调的， partition 最小单位为 cylinder
# 此外，上头显示的那个 Id 为主要档案格式的代号，你可以按下 l ( L 的小写 )
# 就可以知道我们 linux 的 fdisk 认识多少档案系统啰！ ^_^
# 至于 Blocks 则以 KBytes 来显示该 partition 的容量的

Command (m for help): q
# 想要不储存离开吗？按下 q 就对了！不要随便按 w 啊！

范例：查阅目前系统内的所有 partition 有哪些？
[root@linux ~]# fdisk -l
Disk /dev/hda: 20.5 GB, 20520493056 bytes
255 heads, 63 sectors/track, 2494 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1         765     6144831   83  Linux
/dev/hda2             766        1147     3068415   83  Linux
/dev/hda3            1148        1274     1020127+  82  Linux swap
/dev/hda4            1275        2494     9799650    5  Extended
/dev/hda5            1275        2494     9799618+  83  Linux

Disk /dev/hdb: 30.7 GB, 30735581184 bytes
255 heads, 63 sectors/track, 3736 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1   *           1        3633    29182041   83  Linux
# 由于我的这个系统有两颗硬盘，下达 fdisk -l 的话，所有的 partition 都看到了！
```

```
# 另外，我可以确定我的 /dev/hdb 还有剩余空间喔！因为由上面的信息看来，
# 我的 /dev/hdb 的磁柱应该可以到 3736 ，但是目前只用到 3633 ，所以，
# 就肯定还有剩余空间拉！等一下我们就用这个来测试啰！
```

还记得我们刚刚在 认识 EXT2 档案系统 里面提到的 partition 部分内容吗？其实 fdisk 最主要的工作就是在修改『 partition table 』而已，并没有实际的将硬盘切切割割的啦！他会定义出某一个 partition 是由 n1 磁柱到 n2 磁柱之间这样的信息！因此，如果硬盘分割错误时，只要在 format 之前将 partition tables 复原，那么就可以将硬盘原来的数据救回来啰！所以，一个好的管理员，有时候也会将自己的 partition table 记录下来，以备不时之需呀！

这个 fdisk 只有 root 才能执行，此外，请注意， 使用的『装置名称』请不要加上数字，因为 partition 是针对『整个硬盘装置』而不是某个 partition 呢！所以执行 fdisk /dev/hdb1 就会发生错误啦！要使用 fdisk /dev/hdb 才对！那么我们知道可以利用 fdisk 来查阅硬盘的 partition 信息外，底下再来说一说进入 fdisk 之后的几个常做的工作！

Tips:
你可以使用 fdisk 在您的硬盘上面胡搞瞎搞的进行实际操作，都不打紧， 但是请『千万记住，不要按下 w 即可！』离开的时候按下 q 就万事无妨啰！



- 删除磁盘分割槽

刚刚的 fdisk 结果当中，我知道我的 /dev/hdb 仅有 /dev/hdb1 而已， 那么假设我要将这个 /dev/hdb1 删除的话，可以怎么做？

1. fdisk /dev/hdb ：先进入 fdisk 画面；
2. p ：先看一下扇区的信息，假设要杀掉 /dev/hdb1；
3. d ：这个时候会要你选择一个 partition ，就选 1 啰！
4. w (or) q ：按 w 可储存到磁盘数据表中，并离开 fdisk ；当然啰， 如果你反悔了，呵呵，直接按下 q 就可以取消刚刚的删除动作了！

```
[root@linux ~]# fdisk /dev/hdb
1. 先看看整个结果是如何～
Command (m for help): p

Disk /dev/hdb: 30.7 GB, 30735581184 bytes
255 heads, 63 sectors/track, 3736 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1   *           1        3633    29182041   83  Linux

2. 按下 d 给他删除吧！
Command (m for help): d
Selected partition 1
```

```
# 因为我们这个磁盘仅有 1 个 partition ，所以系统会自动帮我们～

Command (m for help): p

Disk /dev/hdb: 30.7 GB, 30735581184 bytes
255 heads, 63 sectors/track, 3736 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
# 『看』不见了！ partition 就这样不见了！

Command (m for help): q
# 鸟哥这里仅是做一个练习而已，所以，按下 q 就能够离开啰～
```

- 新增磁盘分割槽

那么如何新增 partition 呢？以鸟哥刚刚的 /dev/hdb 为例， 我的 /dev/hdb 有一个 /dev/hdb1 ，而且还有剩余空间，那我如何利用？

1. fdisk /dev/hdb ：先进入 fdisk 画面中；
2. n：新增一个扇区，这个时候系统会问你， 如果您已经具有 extended 扇区时，那么系统会问您，您要新增的是 Primary 还是 Logical ，而如果您还没有 extended ，那么系统仅会问你要新增 Primary 还是 Extended。除此之外，如果您已经用完了四个 P＋E 的话，那么就仅有 Logical 可以选择啦！请再回到刚刚说明硬盘的地方再次的复习一下吧！如果是选择 primary 的话，请按 p ，否则请按 e (extended) 或 l (logical)。
3. p ：由于选 择为 primary 所以就会按下 p 啰！
4. 1-4 ：primary 只允许四个， 所以这里请按尚未被使用的那一个扇区啰！
5. w ：同样的储存离开啰！

好了，假设鸟哥想要将我刚刚的 /dev/hdb 剩余空间分为两个分割槽， 一个是 primary ，另一个则是 logical ，且 primary 只要 100MBytes 就够了！ 其它的都分给 logical ，那可以这么做！

```
[root@linux ~]# fdisk /dev/hdb
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p   <==就是这里！可以自行决定是 p 还是 e 喔！
Partition number (1-4): 4  <==编号可以随意！
First cylinder (3634-3736, default 3634):  <==这里按下 Enter 就会使用默认值
Using default value 3634
Last cylinder or +size or +sizeM or +sizeK (3634-3736, default 3736): +100M
# 这个地方有趣了！我们知道 partition 是由 n1 到 n2 的磁柱 (cylinder)，
# 但是我们对于磁柱的大小不容易掌握！这个时候可以填入 +100M 来让系统自动帮我们找出
# 『最接近 100M 的那个 cylinder 号码』！因为不可能刚好等于 100MBytes 啦！
```

```
Command (m for help): p

Disk /dev/hdb: 30.7 GB, 30735581184 bytes
255 heads, 63 sectors/track, 3736 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1   *           1        3633    29182041   83  Linux
/dev/hdb4            3634        3646      104422+  83  Linux
# 这个就是刚刚建立起来的 primary partition 啰！再继续吧！

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
Partition number (1-4): 2
First cylinder (3647-3736, default 3647): <==这里按下 Enter 就会使用默认值
Using default value 3647
Last cylinder or +size or +sizeM or +sizeK (3647-3736, default 3736): <==Enter
Using default value 3736

Command (m for help): p

Disk /dev/hdb: 30.7 GB, 30735581184 bytes
255 heads, 63 sectors/track, 3736 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1   *           1        3633    29182041   83  Linux
/dev/hdb2            3647        3736      722925    5  Extended
/dev/hdb4            3634        3646      104422+  83  Linux
# 仔细看，我们说过， 1-4 号是磁盘保留的号码，所以这个号码可以随意设定，
# 不一定要由 1 开始呢！但是，等一下做的 logical 就一定是由 5 开始累加了！

Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l  <== 使用的是 logical 的 partition 喔！
First cylinder (3647-3736, default 3647):<==Enter
Using default value 3647
```

```
Last cylinder or +size or +sizeM or +sizeK (3647-3736, default 3736):<==Enter
Using default value 3736

Command (m for help): p

Disk /dev/hdb: 30.7 GB, 30735581184 bytes
255 heads, 63 sectors/track, 3736 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1   *           1        3633    29182041   83  Linux
/dev/hdb2            3647        3736      722925    5  Extended
/dev/hdb4            3634        3646      104422+  83  Linux
/dev/hdb5            3647        3736      722893+  83  Linux
# 这可就 OK 啰～虽然新作出三个 partition ，不过仅有 /dev/hdb4 与
# /dev/hdb5 可以用啊！

Command (m for help): w

The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
# 有的时候，磁盘分割表变动之后，得要重新开机，有的则不需要～
# 上面的讯息告诉我们，需要重新开机呢！那就 reboot 吧！
```

透过上面的例子，您可以清楚的看到，呵呵！第一个 logical 是在 5 号呦！OK！在 fdisk 完成之后，请记得使用 mke2fs 格式化啰！另外，请注意，如果过程中进行错误时， 那么赶紧按下 q 离开即可！

- 操作环境的说明

以 root 的身份进行硬盘的 partition 时，最好是在单人维护模式底下比较安全一些， 此外，在进行 fdisk 的时候，如果该硬盘某个 partition 还在使用当中， 那么很有可能系统核心会无法重新加载硬盘的 partition table ，解决的方法就是将该使用中的 partition 给他卸载，然后再重新进入 fdisk 一遍，重新写入 partition table ，那么就可以成功啰！

- 注意事项：

另外，请注意一下，虽然一颗硬盘最大的逻辑扇区可以到达 63 号(总数，包含 1~4 的 primary partition)，但是并非所有的 Linux distribution 都会将所有的逻辑扇区对应的磁盘代号都写入系统当中，以 Red Hat 为例，他仅列出 1~16 个代码，其它的您就得自己动手做啦！至于 Fedora 的话，他则是使用自己侦测的，当您以 fdisk 设定好了 partition table 之后，磁盘对应的磁盘代号就会自动的在您的 /dev/ 里头设定完成啰!不过,有的时候您还是得自己设定一下磁盘代码啦!如何设定呢？就使用 mknod 这个指令吧！

磁盘格式化

```
[root@linux ~]# mke2fs [-bicLj] 装置名称
参数：
-b  ：可以设定每个 block 的大小，目前支持 1024, 2048, 4096 bytes 三种；
-i  ：多少容量给予一个 inode 呢？
-c  ：检查磁盘错误，仅下达一次 -c 时，会进行快速读取测试；
      如果下达两次 -c -c 的话，会测试读写(read-write)，会很慢～
-L  ：后面可以接表头名称 (Label)，这个 label 是有用的喔！后面会讲～
-j  ：本来 mke2fs 是 EXT2 ，加上 -j 后，会主动加入 journal 而成为 EXT3。
范例：
范例一：将刚刚建立的 /dev/hdb5 格式化成为 ext3 吧！且名称为 logical
[root@linux ~]# mke2fs -j -L "logical" /dev/hdb5
mke2fs 1.37 (21-Mar-2005)
Filesystem label=logical
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
90432 inodes, 180723 blocks
9036 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=188743680
6 block groups
32768 blocks per group, 32768 fragments per group
15072 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 27 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
# 这样子就能够将我们的系统给他建立起来啰～

范例二：承上题，如果将 block 改为 2048 ，且 inode 改为 4096?
[root@linux ~]# mke2fs -j -L "logical" -b 2048 -i 4096 /dev/hdb5
# 呈现出来的结果大致与范例一相似～不过就是 block 大小与 inode 数量会改变！
```

这是用来将磁盘格式化成 Linux 系统文件的指令。基本上，只要写入对的装置档案就可以了。 例如我们要格式化软盘的话，或是新的硬盘 /dev/hda5 等等！ 这个指令通常是在新的硬盘上面切割完之后，再加以格式化的！另外，如果要将旧的扇区格式化成 ext2 格式的话，就使用这个指令吧！进行当中显示的讯息有点像上面的最后几行， 系统会显示目前的格式化的默认值！

而如果要设定不同的 Block ，就可以使用 -b 这个参数！请注意啰，预设的情况下， Block 是 4096！此

外，您也可以自订 inode table 呢！而，当没有指定的时候， mke2fs 使用 ext2 为格式化档案格式，若加入 －j 时，则格式化为 ext3 这个 Journaling 的 filesystem 呦！

上面提到的是关于将磁盘给他格式化成为 ext2/ext3 档案系统的指令， 那么如果想要格式化成为其它的档案系统呢？可以直接使用 mkfs 这个指令喔！ 这个指令其实是将几个指令整合的一个功能而已!实际上，你可以参考：『 ls -l /sbin/mkfs* 』来看看系统有的、可以支持的档案格式呢！ 利用 man mkfs 就能够查阅啰！

接下来，如果我想要制作一个可以开机进入 Linux 的软盘片呢？ 可以有底下这个作法喔！

- mkbootdisk (制作软盘开机片)

```
[root@linux ~]# mkbootdisk --device /dev/fd0 `uname -r`
```

这是制作开机磁盘的指令，其中，『 `uname -r` 』是目前 Linux 系统所使用的核心版本， 如果你有多个核心版本的话，你可以直接输入核心版本。例如在鸟哥的系统中， 旧的核心还是有保留的，所以我都会至少有两个核心，在我管理的某部主机中，核心为 2.6.11-1.1369_FC4 及 2.6.12-1.1398_FC4 ，那么如果我要直接以 2.6.11-1.1369_FC4 来开机的话，就可以使用：

mkbootdisk --device /dev/fd0 2.6.11-1.1369_FC4

这个时候， mkbootdisk 就会以 /lib/modules 目录下的数据，配合 /boot 底下的 kernel 档案，来建立可开机的磁盘啰～建立软盘开机片一直是个好主意！ 他可以在你求助无门的时候给你莫大的帮助喔!所以，建立一个新的软盘开机片是一个好主意啦！

- fdformat (进行软盘低阶格式化)

```
[root@linux ~]# fdformat /dev/fd0H1440
```

这是用来『低阶格式化』软盘的指令。(注意：软盘的装置文件为 /dev/fd0)！在上面的装置档案为 /dev/fd0H1440 ， 其中加在 /fd0 之后的 H1440 为表示 1.44MB 的软盘容量！ 在低阶格式化之后，还要将软盘的档案格式化为 Linux 的 ext2 之型态，则需要使用 mke2fs 指令！

磁盘检验： fsck, badblocks

现在也建立好了新的 partition 了，也 format 好了，那么有没有其它的关于硬盘的工作需要来进行呢？有的，就是需要怎样来检查硬盘有没有坏轨呢？那个就是 fsck 这个工具的用途啦！此外，您会发现到，在 / 这个目录底下 (其实只要有挂载硬盘的那个目录底下都有这个目录) 会有一个特殊的目录，就是『 lost+found 』这个目录啦！对的！就是当你处理完 fsck 之后，如果程序有发现到任何的错误的档案，就会将该档案的数据给他丢到这个目录当中，嘿嘿！ 所以当你发现你的 Linux 目录当中有这个档案时，不要担心，那个是正常的啦！而且 只有挂载 partition 的目录 (就是挂载点) 才会有这个预设的目录啰！

还有，由于在 Linux 系统当中，为了增加系统效能，通常系统预设就是一些数据会写在内存当中， 并不会直接将数据写入硬盘里面，这是因为内存的速度要比硬盘快上若干倍呀！ 但是有个问题就发生了，万一系统由于『跳电』或者是其它的莫名原因，造成系统的 shutdown 时，唉呀！怎么办？！系统就完蛋啦！所以啰，我们需要在某些特定的时候让数据直接回存到硬盘之中呀！ 瞭乎！这里提供几个惯用的指令，其中，那个 fsck 是相当重要的，请参考其用法啰！

- fsck

```
[root@linux ~]# fsck [-AtCary] 装置名称
参数：
-t  ：fsck 可以检查好几种不同的 filesystem ，而 fsck 只是一支综合程序而已。
      个别的 filesystem 的检验程序都在 /sbin 底下，您可以使用 ls -l /sbin/fsck*
      去检查看看，就知道有几种 filesystem 啰。预设的 FC4 情况下，至少有：
      ext2, ext3, vfat, msdos 等等 filesystem。
-A  ：依据 /etc/fstab 的内容，将所有的装置都扫瞄一次 (通常开机过程中就会执行此一指令)
-a  ：自动修复检查到的有问题的扇区，所以你不用一直按 y 啰!
-r  ：一定要让使用者决定是否需要修复，这与上一个 -a 刚好相反!
-y  ：与 -a 类似，但是某些 filesystem 仅支持 -y 这个参数，所以您也可以利用 -y 啦!
-C  ：可以在检验的过程当中，使用一个长条图来显示目前的进度!
-f  ：强制检查! 一般来说，如果 fsck 没有发现任何 unclean 的旗标，不会主动进入
      细部检查的，如果您想要强制 fsck 进入细部检查，就得加上 -f 旗标啰!
范例：
范例一：将前面我们建立的 /dev/hdb5 这个装置给他检验一下!
[root@linux ~]# fsck -C -t ext3 /dev/hdb5
fsck 1.37 (21-Mar-2005)
e2fsck 1.37 (21-Mar-2005)
logical: clean, 11/181056 files, 21706/361446 blocks
# 如果一切没有问题，就会出现上述的讯息~
```

这是用来检查与修正硬盘错误的指令。注意：通常只有身为 root 且你的系统有问题的时候才使用这个指令，否则在正常状况下使用此一指令， 可能会造成对档案的危害！通常使用这个指令的场合都是在系统出现极大的问题，导致你在 Linux 开机的时候得进入单人单机模式下进行维护的行为时，才必须使用此一指令！ 另外，如果你怀疑刚刚格式化成功的硬盘有问题的时后，也可以使用 fsck 来检查一下硬盘呦！其实就有点像是 Windows 的 scandisk 啦！此外，由于 fsck 在扫瞄硬盘的时候，可能会造成部分 filesystem 的损坏，所以『执行 fsck 时， 被检查的 partition 务必不可挂载到系统上！亦即是需要在卸载的状态喔！』

常常我们会发现，在比较老旧的机器上 (例如鸟哥的 p-166)，如果主机不正常的关机 (例如跳电啰！)，那么硬盘很可能会出现错误的状况！这个时候 Linux 就无法正常的开机！这个时候就需要输入 root 的密码，以登入单人维护模式 (run level 1)，然后下达 fsck -y /dev/hdxxx 来检查你的硬盘！ 等到确认成功之后，就使用 reboot 来重新启动吧！

- badblocks

```
[root@linux ~]# badblocks -[svw] 装置名称
参数：
-s  ：在屏幕上列出进度
-v  ：可以在屏幕上看到进度
-w  ：使用写入的方式来测试，建议不要使用此一参数，尤其是待检查的装置已有档案时!
范例：
[root@linux ~]# badblocks -sv /dev/hdb5
Checking blocks 0 to 722893
Checking for bad blocks (read-only test): done                        893
```

```
Pass completed, 0 bad blocks found.
```

这是用来检查硬盘或软盘扇区有没有坏轨的指令！跟 Windows 的 scandisk 相同功能啦！不过由于 fsck 的功能比较强，所以目前大多已经不使用这个指令了！

- sync

在正常的状况中，由于为了增加系统的效率，因此， 很多时候进行中的程序产生的程序之临时文件都不会直接存至磁盘驱动器当中， 而是记忆在内存当中！由于内存的数据传递速度比磁盘驱动器快了几十倍， 所以如此一来将有助于整个系统的效率！！然而这也产生了一个困扰， 那就是当你的系统不正常关机的时候，可能会使得一些已经经过改变，却还没有存入磁盘中的数据遗失(因为还在内存当中!)所以这个时候 sync 的功能就相当的大了！ 因为他可以直接将系统暂存在内存当中的数据回存写入磁盘当中，呵呵！ 很棒吧！但是需要注意你的系统核心 (kernel) 必须要有支持 sync 才行（目前几乎一定都会支持的啦！）

磁盘挂载与卸载

要将上面我们所建立起来的磁盘档案系统或软盘正式的在 Linux 上面启用时，一定需要将他挂载上档案系统！ 而所谓的『挂载点』则是该 partition 所在的目录，且在该目录下的所有目录都归在该 partition 所有！假设一个情况好了，我们的 / 为 /dev/hda1 而 /home 为 /dev/hda2 ，那么在 /home/test 底下的咚咚就也都归 /dev/hda2 这个 partition 所有啰！而需要特别留意的是，由于挂载档案系统需要挂载点，所以挂载的时候得先建立起挂载的目录才行！

除此之外，如果您要用来挂载的目录里面并不是空的，那么挂载了档案系统之后， 那么原目录下的东西就会暂时的消失。举个例子来说，假设您的 /home 原本是属于根目录 / 底下的 partition 所有，底下原本就有 /home/test 与 /home/vbird 两个目录。然后你想要加入新的硬盘，并且直接挂载 /home 底下，那么当您挂载上新的 partition 时，则 /home 目录显示的是该 partition 的内容，至于原先的 test 与 vbird 这两个目录就会暂时的被隐藏掉了！注意喔！并不是被覆盖掉， 而是暂时的隐藏了起来，等到 partition 被 umount 之后， 则该目录的内容就会再次的跑出来啦！

而要将档案系统挂载到我们的 Linux 系统上，就要使用 mount 这个指令啦！ 不过，这个指令真的是博大精深～粉难啦！我们学简单一点啊～ ^_^

```
[root@linux ~]# mount -a
[root@linux ~]# mount [-tonL]  装置名称代号  挂载点
参数：
-a  ：依照 /etc/fstab 的内容将所有相关的磁盘都挂上来！
-n  ：一般来说，当我们挂载档案系统到 Linux 上头时， Linux 会主动的将
      目前的 partition 与 filesystem 还有对应的挂载点，都记录到 /etc/mtab
      那个档案中。不过，有些时刻 (例如不正常关机导致一些问题，而进入单人模式)
      系统无法写入 /etc/mtab 时，就可以加上 -n 这个参数来略过写入 mtab 的动作。
-L  ：系统除了利用装置名称代号 (例如 /dev/hda1) 之外，还可以利用 partition
      的表头名称 ( Label ) 来进行挂载喔！所以，最好为您的 partition 取一个
      在您系统当中独一无二的名称吧！
-t  ：您的 Linux 支持的档案格式，就写在这里吧！举例来说，我们在上面建立
      /dev/hdb5 是 ext3 档案系统，那么要挂载时，就得要加上 -t ext3
      来告知系统，用 ext3 的档案格式来挂载该 partition 呢！
```

```
      至于系统支持的 filesystem 类型在 /lib/modules/`uname -r`/kernel/fs 当中。
      常见的有：
         ext2, ext3, reiserfs, 等 Linux 惯用 filesystem
         vfat, msdos 等 Windows 常见 filesystem
         iso9660 为光盘片的格式
         nfs, smbfs 等为网络相关档案系统。这部分未来我们会在网络方面提及！
      若 mount 后面没有加 -t 档案系统格式时，则 Linux 在预设的情况下，
      会主动以 /etc/filesystems 这个档案内规范的档案系统格式
      来尝试主动的挂载喔！
-o  ：后面可以接一些挂载时，额外加上的参数喔！比方说账号、密码、读写权限等：
      ro, rw:       此 partition 为只读(ro) 或可擦写(rw)
      async, sync:  此 partition 为同步写入 (sync) 或异步 (async)，这个与我们
                    之前提到的档案系统运作方式有关，预设是 async
      auto, noauto: 允许此 partition 被以 mount -a 自动挂载(auto)
      dev, nodev:   是否允许此 partition 上，可建立装置档案？ dev 为可允许
      suid, nosuid: 是否允许此 partition 含有 suid/sgid 的档案格式？
      exec, noexec: 是否允许此 partition 上拥有可执行 binary 档案？
      user, nouser: 是否允许此 partition 让 user 执行 mount ？一般来说，
                    mount 仅有 root 可以进行，但下达 user 参数，则可让
                    一般 user 也能够对此 partition 进行 mount 。
      defaults:     默认值为：rw, suid, dev, exec, auto, nouser, and async
      remount:      重新挂载，这在系统出错，或重新更新参数时，很有用！
范例：
范例一：将刚刚建立的 /dev/hdb5 挂载到 /mnt/hdb5 上面！
[root@linux ~]# mkdir /mnt/hdb5
[root@linux ~]# mount -t ext3 /dev/hdb5 /mnt/hdb5
[root@linux ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
....中间省略.....
/dev/hdb5               700144     20664    643336   4% /mnt/hdb5

范例二：挂载光盘！
[root@linux ~]# mount -t iso9660 /dev/cdrom /mnt/cdrom
[root@linux ~]# mount /dev/cdrom
# 上面的参数当中提到，如果没有加上 -t 这个参数时，系统会主动的以
# /etc/filesystems 里面规范的内容给他测试一下是否挂载～另外，
# 因为我们的 /etc/fstab 里面会规范 /dev/cdrom 应该挂载到那个挂载点，
# 因此，直接下达 mount /dev/cdrom 也是可以的喔！(当然要看/etc/fstab 设定啦！)

范例三：挂载 Window fat 软盘！
[root@linux ~]# mount -t vfat /dev/fd0 /mnt/floppy

范例四：将 / 重新挂载，并加入参数为 rw ！
```

```
[root@linux ~]# mount -o remount,rw /

范例五：将 Label 名为 logical 的 partition 挂载到 /mnt/hdb5 中
[root@linux ~]# mount -t ext3 -L logical /mnt/hdb5

范例六：将系统所有的以挂载的 partition 数据列出来
[root@linux ~]# mount
/dev/hda1 on / type ext3 (rw)
/dev/proc on /proc type proc (rw)
/dev/shm on /dev/shm type tmpfs (rw)
/dev/hda5 on /home type ext3 (rw)
/dev/hdb5 on /mnt/hdb5 type ext3 (rw)
# 嗯！不加任何参数，则 mount 会将目前系统的所有 partition
# 与相关对应的 filesystem 及 mount point 都列出来!
```

在预设的情况下， mount 这个指令只有 root 才能执行！ 如果您想要将整个系统里面记录的 filesystem 与 mount point 对应的数据（记录在 /etc/fstab 文件中！）， 全部都挂载上来，那么请执行：

mount -a

就可以依照 /etc/fstab 的参数内容将所有的磁盘给他重新挂上去！此外，需要注意的是，由于 Linux 系统中，每一个路径都有可能是一个独立的扇区系统， 所以需要将每个扇区系统都挂上各自的挂载点！详细的内容请回去参考一下上一篇 Linux 档案系统 的说明。另外， 由于各个扇区的档案系统可能并不相同，所以您必须先要了解该扇区的档案系统，这样才可以进行 mount 的工作!如何知道该磁盘的档案格式呢？可以使用 fdisk 来显示的功能即可！

另外，如果您没有加上 -t 的参数，那么系统会预设尝试以 /etc/filesystems 内的档案系统格式来测试一下是否可以将装置挂载上来呢！

请注意呦！由于 mount 之后的 partition 就已经被设定在使用了， 所以，您不可以使用 fsck 检查该 partition 呢!否则可能会造成 filesystem 的损毁～ 因此,你就必须要将该 partition 给卸载才行！可以利用 umount 来卸载喔！

另外，我们也可以利用 mount 来将某个目录挂载到另外一个目录去喔！ 这并不是挂载档案系统，而是额外挂载某个目录的方法！ 其实可以利用 link file 来达到底下范例的功能啦！ ^_^

```
范例一：将 /home 这个目录暂时挂载到 /tmp/home 底下：
[root@linux ~]# mkdir /tmp/home
[root@linux ~]# mount --bind /home /tmp/home
[root@linux ~]# ls -lid /home/ /tmp/home
159841 drwxr-xr-x  6 root root 4096 May 30 20:07 /home/
159841 drwxr-xr-x  6 root root 4096 May 30 20:07 /tmp/home

范例二：将 /tmp/home 卸载：
[root@linux ~]# umount /tmp/home
```

看起来，其实两者连结到同一个 inode 嘛！！ ^_^ 没错啦！透过这个 mount --bind 的功能， 您可以将某个目录挂载到其它目录去喔！而并不是整块 filesystem 的啦！

- umount (将装置档案卸载)

```
[root@linux ~]# umount 装置代号或挂载点
[root@linux ~]# umount /dev/hdb5
[root@linux ~]# umount /mnt/hdb5
```

就是直接将 mount 上来的档案系统给他卸载即是！卸载之后，可以使用 df 看看是否还存在呢？！ 此外，也可以利用 -f 参数将想要卸载的 partition 强制卸载！此外， 卸载的方式，可以下达装置 (如 /dev/hdb5) 或挂载点 (如 /mnt/hdb5)，均可接受啦！

磁盘参数修订：

某些时刻，您可能会希望修改一下目前磁盘的一些相关信息，举例来说，磁盘的 Label ， 或者是 journal 的参数，或者是其它硬盘运作时的相关参数 (例如 DMA 启动与否～)。 这个时候，就得需要底下这些相关的指令功能啰～

- mknod

```
[root@linux ~]# mknod 装置名称 [bcp] [Major] [Minor]
参数：
装置种类：
   b  ：设定装置名称成为一个周边储存设备档案，例如硬盘等；
   c  ：设定装置名称成为一个周边输入设备档案，例如鼠标/键盘等；
   p  ：设定装置名称成为一个 FIFO 档案；
Major ：主要装置代码；
Minor ：次要装置代码；
范例：
范例一：建立 /dev/hda10 这个磁盘储存装置
[root@linux ~]# mknod /dev/hda10 b 3 10
# 上面那个 3 与 10 是有意义的，不要随意设定啊！
```

还记得我们说过，在 Linux 底下所有的装置都以档案来代表吧！？ 但是那个档案如何代表该装置呢？很简单！就是透过档案的 major 与 minor 数值来替代的～所以，那个 major 与 minor 数值是有特殊意义的， 不是随意设定的喔！举例来说，如果以硬盘装置来说明，那么 /dev/hda 到 /dev/hdd 的 major 与 minor 代码是：

硬盘代号	Major	Minor
/dev/hda	3	0~63
/dev/hdb	3	64~127
/dev/hdc	22	0~63
/dev/hdd	22	64~127

此外， mknod 也可以用来制作 FIFO 类型的档案喔！更多与 Linux 核心有关的装置及装置代号可以参考：

- http://www.kernel.org/pub/linux/docs/device-list/devices.txt

* e2label

```
[root@linux ~]# e2label 装置名称  新的 Label 名称
[root@linux ~]# e2label /dev/hdb5 hdb5
[root@linux ~]# dumpe2fs -h /dev/hdb5
Filesystem volume name:   hdb5
.....其它省略.....
```

这个东西也挺有趣的，主要的功能是用来修改『磁盘的表头数据』，也就是 label 啦。 那是甚么东西呢？如果你使用过 Windows 的档案总管的话，那么应该会晓得， 每个磁盘驱动器代号后面都有个名称吧，呵呵！那个就是 label 啰。这个东西除了有趣之外， 也会被使用到一些设定档案当中，举例来说，当我们在挂载磁盘的时候，除了利用磁盘的代号之外 (/dev/hdxx) 也可以直接利用磁盘的 label 来作为挂载的磁盘挂载点喔！基本上， 就是那个 /etc/fstab 档案的设定啰！

因为某些 distribution 为了方便，他们是以 Label 来做为磁盘挂载的依据， 这样有好有坏啦！

* 优点：不论硬盘代号怎么变，不论您将硬盘插在那个 IDE 接口 (IDE1 或 IDE2 或 master 或 slave 等)，由于系统是透过 Label ，所以，磁盘插在那个接口将不会有影响。
* 缺点：如果插了两颗硬盘，刚好两颗硬盘的 Label 有重复的，那就惨了～ 因为系统会无法判断那个磁盘分割槽才是正确的！

所以，鸟哥通常还是比较喜欢直接利用磁盘代号来挂载啦！ 不过，如果没有特殊需求的话，那么利用 Label 来挂载也成！ 但是您就不可以随意修改 Label 的名称了！

* tune2fs

```
[root@linux ~]# tune2fs [-jlL] 装置代号
参数：
-j  ：将 ext2 的 filesystem 转换为 ext3 的档案系统；
-l  ：类似 dumpe2fs -h 的功能～将 superblock 内的数据读出来～
-L  ：类似 e2label 的功能，可以修改 filesystem 的 Label 喔！
范例：
[root@linux ~]# tune2fs -l /dev/hdb5
```

这个指令的功能其实很广泛啦～上面鸟哥仅列出很简单的一些参数而已， 更多的用法请自行参考 man tune2fs 。比较有趣的是，如果您的某个 partition 原本是 ext2 的档案系统，如果想要将他更新成为 ext3 档案系统的话， 利用 tune2fs 就可以很简单的转换过来啰～

* hdparm

如果您的硬盘是有 DMA 模式功能的，但是系统却没有启动他，那么您的硬盘存取效能可能会降低一半以上～ 所以，当然要启动 DMA 啦～那么如何启动？就用 hdparm 啊！ 不过， hdparm 里头有很多很进阶的参数设定值，一般来说， 不很建议大家随意修订～很容易造成硬盘的挂点喔！ 用这个指令时，最多是启动 DMA 模式，以及测试硬盘的存取效能就好了～ 真的不要随意更动其它参数喔！除非您真的知道自己在干嘛～

```
[root@linux ~]# hdparm [-icdmXTt] 装置名称
参数：
```

```
-i  ：系统在开机的过程当中，会利用本身核心的驱动程序(模块)来测试硬盘，
      利用 -i 参数，可将这些测试值取出来，这些值不一定是正确的，
      不过，却可以提供我们一个参考值的依据！
-c  ：设定 32-bit (32 位)存取模式。这个 32 位存取模式指的是在硬盘在与
      PCI 接口之间传输的模式，而硬盘本身是依旧以 16 位模式在跑得！
      预设的情况下，这个设定值都会被打开，建议直接使用 c1 即可！
-d  ：设定是否启用 dma 模式， -d1 为启动， -d0 为取消；
-m  ：设定同步读取多个 sector 的模式。一般来说，设定此模式，可降低系统因为
      读取磁盘而损耗的效能～不过， WD 的硬盘则不怎么建议设定此值～
      一般来说，设定为 16/32 是最佳化，不过，WD 硬盘建议值则是 4/8 。
      这个值的最大值，可以利用 hdparm -i /dev/hda 输出的 MaxMultSect
      来设定喔！一般如果不晓得，设定 16 是合理的！
-X  ：设定 UtraDMA 的模式，一般来说， UDMA 的模式值加 64 即为设定值。
      并且，硬盘与主机板芯片必须要同步，所以，取最小的那个。一般来说：
      33 MHz DMA mode 0~2 (X64~X66)
      66 MHz DMA mode 3~4 (X67~X68)
      100MHz DMA mode 5   (X69)
      如果您的硬盘上面显示的是 UATA 100 以上的，那么设定 X69 也不错！
-T  ：测试暂存区 cache 的存取效能
-t  ：测试硬盘的实际存取效能 （较正确！）
范例：
范例一：取得我硬盘的最大同步存取 sector 值与目前的 UDMA 模式
[root@linux ~]# hdparm -i /dev/hda
 Model=ST320430A, FwRev=3.07, SerialNo=7BX02236
 Config={ HardSect NotMFM HdSw>15uSec Fixed DTR>10Mbs RotSpdTol>.5% }
 RawCHS=16383/16/63, TrkSize=0, SectSize=0, ECCbytes=0
 BuffType=unknown, BuffSize=512kB, MaxMultSect=16, MultSect=16
 CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAsects=40079088
 IORDY=on/off, tPIO={min:240,w/IORDY:120}, tDMA={min:120,rec:120}
 PIO modes:  pio0 pio1 pio2 pio3 pio4
 DMA modes:  mdma0 mdma1 mdma2
 UDMA modes: udma0 udma1 udma2 udma3 *udma4
 AdvancedPM=no WriteCache=enabled
 Drive conforms to: device does not report version:  1 2 3 4
# 在输出的数据中，有几个比较重要的，除了 MaxMultSec 这个数值外，
# 那个 UDMA modes: 会显示目前的 UDMA 模式 (有 * 号的那个为目前的值)。

范例二：取得我主机板上面关于 IDE 的速度限制
[root@linux ~]# lspci -v
00:07.1 IDE interface: VIA Technologies, Inc. VT82C586/B/686A/B PIPC Bus Master
IDE (rev 10) (prog-if 8a [Master SecP PriP])
        Subsystem: VIA Technologies, Inc. VT8235 Bus Master ATA133/100/66/33 IDE
        Flags: bus master, medium devsel, latency 32
```

```
        I/O ports at d000 [size=16]
        Capabilities: [c0] Power Management version 2
# 我可以透过 lspci 来直接取得 PCI 接口上的各个装置设备。
# 其中，可以找到 IDE 接口，并从中找到关于这个接口可接受的速度呢！

范例三：启动我的 UDMA 在 mode 4 喔～
[root@linux ~]# hdparm -d1 -c1 -X68 /dev/hda
# 由范例一与范例二，鸟哥的主机板上面大概仅能支持到 UDMA 66 吧～
# 那就是 mode4 啰～所以， X = 64+4 = 68 ，因此，设定就是 -X68 啰～

范例四：测试这颗硬盘的读取效能
[root@linux ~]# hdparm -Tt /dev/hda
/dev/hda:
 Timing cached reads:   544 MB in  2.01 seconds = 270.28 MB/sec
 Timing buffered disk reads:   80 MB in  3.01 seconds =  26.56 MB/sec
# 我的机子没有很好啦～这样的速度.....差强人意～
```

我们都知道目前的 IDE 硬盘主要的传输模式为 ATA 模式，最近 （2002 年）已经出到了 ATA 133 了！不过，传统上， ATA 66 就已经很厉害了！新的 IDE 硬盘也没有办法完全利用 ATA 100 呢！但是，你知道吗？有没有开启 ATA 这个传输信道的 DMA 模式， 对于硬盘的存取效率差很大喲！

这个指令必须要 root 才能执行！此外，需要特别注意， -X 这个参数是很危险的参数设定， 除非您非常了解您的硬盘架构，否则不建议自行设定，因为 manual page 上面就有说明到这一点～～不过，无论如何，目前大于 2GB 以上的硬盘至少都已经支持了 DMA 33 了！因此，使用 -X66 应该都是可行的！而如果您的硬盘是很新的， 那么 -X69 应该是没有问题才是！不过，还是要 hdparm -i /dev/hd[a-d] 去检查看看！

设定开机挂载：

在上一小节里面，我们提到了硬盘的分割与格式化，同时还提到了如何挂载的问题等等， 在这个小节当中，我们就持续的来讨论，那么 mount 还可以做哪些事情呢？ 还有，如果想要一开机就让系统自动的帮我们将 partition 挂载起来，又该如何呢？

各式磁盘挂载与 中文编码挂载还有 USB 随身碟：

这里再次强调一个观念，在 Windows 底下，磁盘分割是以 A, B, C, D, ... 等等的方式来划分的，然而在 Linux 或 Unix 系统之下，却是以目录来代表，也就是说， 一个目录很可能就是一个扇区了！ 举个例子来说，通常 Linux 预设的软盘挂载的地点在 /mnt/floppy 这里！呵呵！那么如果你需要软盘的数据时，就将 /dev/fd0 这一个装置(前面提过啰!这个是周边存取装置的一个设备档案类型)挂上 /mnt/floppy 就可以啦！然后你进入到 /mnt/floppy 就可以读取软盘的数据啰！

- 挂载软盘

很多朋友常常会使用到软盘，尤其是在网络有问题的时候，那么如何使用 Linux 将软盘挂上来呢？！首先，您给先了解你的磁盘档案的格式，例如 Linux 的档案格式（ ext2 ）与 Windows 的档案格式（ vfat ）是不一样的！分别可以使用如下的方式挂上来：

```
[root@linux ~]# mount -t ext2 /dev/fd0 /media/floppy
[root@linux ~]# mount -t vfat /dev/fd0 /media/floppy
[root@linux ~]# umount /media/floppy
```

所以啰！即使是 Windows 格式的档案，在 Linux 底下仍然是可以读取的到的呦！另外， 要注意的是，即使你使用软盘完毕之后，一定要将 /media/floppy 卸载之后才可以取出软盘片喔！不然系统会一直告诉你发生错误啰！而在卸载 /media/floppy 的时候，你一定不能在该目录底下，否则会发生错误讯息喔！ 而如果加载的格式不符合，系统也不会将该软盘挂上来的呦！好了，那么怎么制作 ext2 的软盘呢？简单的很，就使用 mke2fs /dev/fd0 就行啦！

- 挂载 Windows 磁盘

如果万一你在安装系统的时候忘记将 Windows 的 VFAT 格式的扇区 mount 上你的 Linux 时，该怎么办？！这种现象最常发生在多系统共存的环境中！例如在原有的 Windows 98 或 Win2000 上面安装 Linux ，但是结果却忘记将该扇区挂载上来！嗯！这样需要重新安装吗？！ 当然不需要了！又不是被入侵！那么该如何挂载上来呢？！

就如同前面说的，由于一个目录可能代表了一个扇区，因此你必须要先建立一个目录， 然后再将此扇区挂载上你的 Linux 目录，就可以啦！另外，由于需要将扇区挂在目录下， 所以你还得需要了解你的 Windows 扇区是在哪一个硬盘周边中喔！如何知道你的 Windows 扇区呢？就使用 fdisk 吧！使用 fdisk -l 就能够知道啰～

那么假设我的 windows 的 VFAT filesystem 是在 /dev/hda1 ，而我想要将该 partition 挂载到 /mnt/win98 ，该如何做？

```
[root@linux ~]# mkdir /mnt/win98
[root@linux ~]# mount -t vfat /dev/hda1 /mnt/win98
[root@linux ~]# mount -t vfat -o iocharset=cp950 /dev/hda1 /mnt/win98
```

如此一来就将 Windows 的系统挂到 Linux 里面啰！简单吧！请注意， 上面那个第三行所使用的参数可以让您的 Windows 98 的扇区当中的档案显示出正确的中文呦！ 因为加入了中文编码啊！ ^_^

- 挂载 USB 随身碟

以现在的科技来说(2005 年)，软盘与光盘不再是最佳的携带工具了～ 最佳的携带储存设备应该是 USB 随身碟或者是随身硬盘～那么我们可以挂载 USB 随身碟吗？！呵呵！当然可以啊～不过.....要我们的 Linux 捉的到 USB 才行～ 不过，您不需要担心，因为，目前的 distribution 均会主动的加载 USB 的模块， 所以，您只要插入 USB 随身碟，嘿嘿！我们的 Linux 几乎没有问题，一定可以捉到的！

捉到 USB 随身碟之后，再利用 fdisk -l 列出所有的 partition ，您会发现， 系统中怎么会多出 /dev/sda[??] 的 SCSI 硬盘啊？！不会吧！系统这么好？？ 呵呵！不是啦～其实 USB 硬盘的代号也是 /dev/sd[a-??] 的代号， 第一个 USB 代号为 /dev/sda ，而如果该 USB 硬盘还有 partition 的话， 那就会有一些号码出现了～如果是随身碟，通常只有 /dev/sda1 而已啦～ 好，那就假设您的随身碟是 /dev/sda1 好了，那么将他挂载到 /mnt/usb ，要怎么做？

```
[root@linux ~]# mkdir /mnt/usb
[root@linux ~]# mount -t vfat /dev/sda1 /mnt/usb
```

上头是假设您的 USB 随身碟使用的是 FAT 的 Windows 档案格式而设定的。 如果您的随身碟是 NTFS 的 Windows 2000 档案格式，那就比较麻烦， 因为 FC4 预设情况下，并不支持这个档案系统的～ 如果您执意要挂载 NTFS 的档案格式，那么..... 请参考底下这个计划的网站啰～

- Linux-NTFS Project: http://linux-ntfs.sourceforge.net/

开机挂载 /etc/fstab 及 /etc/mtab

刚刚上面说了许多，那么可不可以在开机的时候就将我要的扇区都挂好呢？！这样我就不需要每次进入 Linux 系统都还要在挂载一次呀！当然可以啰！那就直接到 /etc/fstab 里面去修修就行啰！不过，在开始说明前，这里要先跟大家说一说系统挂载的一些限制：

- 根目录 / 是必须挂载的，而且一定要先于其它 mount point 被挂载进来。
- 其它 mount point 必须为已建立的目录，可任意指定，但一定要遵守必须的系统目录架构原则
- 所有 mount point 在同一时间之内，只能挂载一次。
- 所有 partition 在同一时间之内，只能挂载一次。
- 如若进行卸载，您必须先将工作目录移到 mount point(及其子目录) 之外。

好了，那么我们进入 /etc/fstab 看一看吧：

```
[root@linux ~]# cat /etc/fstab
# Device    Mount point    filesystem parameters  dump fsck
LABEL=/     /               ext3    defaults        1 1
/dev/hda5   /home           ext3    defaults        1 2
/dev/hda3   swap            swap    defaults        0 0
/dev/hdc    /media/cdrom    auto    pamconsole,exec,noauto,managed 0 0
/dev/devpts /dev/pts        devpts  gid=5,mode=620  0 0
/dev/shm    /dev/shm        tmpfs   defaults        0 0
/dev/proc   /proc           proc    defaults        0 0
/dev/sys    /sys            sysfs   defaults        0 0
```

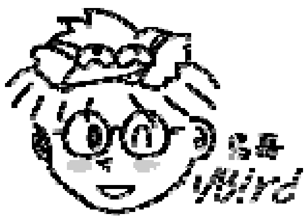
其实这个 /etc/fstab 就是将我们使用 mount 来挂载一个装置到系统的某个挂载点， 所需要下达的指令内容，将这些内容通通写到 /etc/fstab 里面去，而让系统一开机就主动挂载啰～ 那么 mount 下达指令时，需要哪些参数？不就是『装置代号、挂载点、档案系统类别、参数』等等， 而我们的 /etc/fstab 则加入了两项额外的功能，分别是备份指令 dump 的执行与否， 与是否开机进行 fsck 扫瞄磁盘呢～

我这个人比较龟毛一点，因为某些 distributions 的 /etc/fstab 档案排列方式蛮丑的， 虽然每一栏之间只要以空格符分开即可，但就是觉得丑，所以通常鸟哥就会自己排列整齐， 并加上批注符号，就是 # 字号，来帮我记忆这些信息！由上面的说明，我们知道 /etc/fstab 内总共有六栏，分别来谈一谈每一栏的内容吧！

1. 磁盘装置代号或该装置的 Label：
   这个就是装置代号啦！将您需要的装置代号给他填上去吧！！ 不过，还记得我们的 filesystem 可以拥有标头名称吧 (Label)？ 没错，我们也可以利用 Label 来挂载档案系统喔！例如上表当中的特殊字体的第一行， 我的根目录 (/) 就是以 Label 名称为 / 的磁盘分割槽来挂载的啊！ 利

用 label 挂载时，您必须要知道您的磁盘内的 label 名称，可以利用 dumpe2fs 来读取， 也可以利用 e2label 来更改标头名称啊。 在知道了 label 名称后，最后就可以利用 LABEL=(your label name) 来设定您的装置啰～

Tips:
记得有一次有个网友写信给鸟哥，他说，依照 e2label 的设定去练习修改自己的 partition 的 Label name 之后，却发现，再也无法顺利开机成功！ 后来才发现，原来他的 /etc/fstab 就是以 Label name 去挂载的。但是因为在练习的时候，将 Label name 改名字过了，导致无法在开机的过程当中顺利搜寻到～所以啦， 各位亲爱的朋友，这里再次的强调，利用装置名称 (ex> /dev/hda1) 来挂载 partition 时， 虽然是被固定死的，所以您的硬盘不可以随意插在任意的插槽，不过他还是有好处的。 而使用 Label name 来挂载，虽然就没有插槽方面的问题，不过，您就得要随时注意您的 Label name 喔！尤其是新增硬盘的时候！ ^_^



2. 挂载点 (mount point)：：
   就是挂载点啊！挂载点是什么？一定是目录啊～要知道啊！！！

3. 磁盘分割槽的档案系统：
   就如同我们在这个章节一开始就谈到的， Linux 在传统上面，使用的是 ext2/ext3 等档案系统，目前则加入了很多日志式档案系统，例如 reiserfs 及 XFS 等档案系统的支持。 此外，存在已久的 Windows vfat, msdos 及 iso9660 的光盘档案系统， 还有网络档案系统如 nfs, smbfs 等等，都可以被支持。这个字段就是写这些档案系统的地方啊！

4. 档案系统参数：
   每个档案系统还有很多参数可以加入的，例如中文编码的 iocharset=big5,codepage=950 之类的，当然还有很多常见的参数， 虽然之前在 mount 已经提过一次，这里我们利用表格的方式再次的说明一下：

参数	内容意义
async/sync 异步/同步	是否允许磁盘与内存中的数据以同步写入的动作？使用 async 这个异步写入的方式会比较快速一些。
auto/noauto 自动/非自动	在开机的时候是否自动挂载该扇区？既然设定在这个区域内了，当然希望开机的时候自动挂载啰！
rw/ro 可擦写/只读	让该扇区以可擦写或者是只读的型态挂载上来，如果是 vfat 之类的非 Linux 传统扇区，您不想让 Linux 变更的话，那么使用 ro 也不错！能够提供有效的保护呢！
exec/noexec 可执行/不可执行	限制在此档案系统内是否可以进行『执行』的工作？如果是纯粹用来储存资料的， 那么可以设定为 noexec 会比较安全，相对的，会比较麻烦！
user/nouser	是否允许使用者使用 mount 指令来挂载呢？一般而言，我们当然

允许/不允许使用者挂载	不希望一般身份的 user 能使用 mount 啰，因为太不安全了，因此这里应该要设定为 nouser 啰！
suid/nosuid 具有/不具有 suid 权限	该档案系统是否允许 SUID 的存在？一般而言，如果不是 Linux 系统的扇区，而是一般数据的 partition ，那么设定为 nosuid 确实比较安全一些！毕竟有 SUID 是蛮可怕的一件事。
usrquota	注意名称是『 usrquota 』不要拼错了！这个是在启动 filesystem 支持磁盘配额模式，更多数据我们在第四篇再谈。
grpquota	注意名称是『grpquota』，启动 filesystem 对群组磁盘配额模式的支持。
defaults	同时具有 rw, suid, dev, exec, auto, nouser, async 等参数。基本上，预设情况使用 defaults 设定即可！

5.
6. 能否被 dump 备份指令作用：
   在 Linux 当中，可以利用 dump 这个指令来进行系统的备份的。而 dump 指令则会针对 /etc/fstab 的设定值，去选择是否要将该 partition 进行备份的动作呢！ 0 代表不要做 dump 备份， 1 代表要进行 dump 的动作。 2 也代表要做 dump 备份动作， 不过，该 partition 重要度比 1 小。

7. 是否以 fsck 检验扇区：
   开机的过程中，系统预设会以 fsck 检验我们的 partition 内的 filesystem 是否完整 (clean)。 不过，某些 filesystem 是不需要检验的，例如虚拟内存 swap ，或者是特殊档案系统， 例如 /proc 与 /sys 等等。所以，在这个字段中，我们可以设定是否要以 fsck 检验该 filesystem 喔。 0 是不要检验， 1 是要检验， 2 也是要检验，不过 1 会比较早被检验啦！ 一般来说，根目录设定为 1 ，其它的要检验的 filesystem 都设定为 2 就好了。

所以说，如果我想要将我们刚刚练习时，建立的 /dev/hdb5 这个 ext3 的 filesystem 挂载到 /mnt/hdb5 时， 并且在开机的时候就已经自动的挂载好，那么就可以将底下这一行写入到 /etc/fstab 当中了：

```
/dev/hdb5   /mnt/hdb5   ext3    defaults   2   2
```

很简单吧！所以啦，以后您自己建立的磁盘档案系统想在开机的时候挂载好时，就在 /etc/fstab 加入吧！此外，这个 /etc/fstab 还有什么特殊功能呢？还记得使用 mount -a 时，我们提到的该参数参考档案吧？！没错啊！就是这个 /etc/fstab 啊！而且，一般来说， 当我们编辑 /etc/fstab 后，为了避免可能的错误，通常就会以 mount -a 这个指令来测试看看呢！这是很重要的一个测试动作喔！

另外，您也必须要了解到，除了这些磁盘档案格式之外， 其实在系统里面还有一些特殊的格式可以挂载来帮助系统的运作的！ 例如上表当中非特殊字体的那几行字！

而 /etc/fstab 是开机时的设定档，不过，实际 filesystem 的挂载是记录到 /etc/mtab 与 /proc/mounts 这两个档案当中的。每次我们在更动 filesystem 的挂载时，也会同时更动这两个档案喔！但是，万一发生您在 /etc/fstab 输入的数据错误，导致无法顺利开机成功，而进入单人维护模式当中，那时候的 / 可

是 read only 的状态，当然您就无法修改 /etc/fstab ，也无法更新 /etc/mtab 啰～那怎么办？ 没关系，可以利用底下这一招：

```
[root@linux ~]# mount -n -o remount,rw /
```

加上 -n 则不更新 /etc/mtab ，加上 -o 则提供额外的参数设定。 利用这一动作，嘿嘿！您的 / 就可以读写，那么自然就能够更新档案内容啰～

特殊装置 loop 挂载

除了常见的软、硬盘挂载之外，我们还可以挂载特殊装置喔！举例来说，利用我们的硬盘内的档案仿真出来的装置！ 也就是说，当我的硬盘内有一个 2GB 的档案时，我可以将这个档案『模拟』成为一个独立的装置， 然后用这个装置来挂载使用喔！当然啦，这个 2GB 的大档案要能够被挂载时， 他必须是一个『被格式化过的档案』才行！底下我们就来玩一玩这个咚咚。

建立大型档案

首先，我们得先有一个大的档案吧！怎么建立这个大档案呢？在 Linux 底下我们有一支很好用的程序 dd 可以用来建立空的档案喔！详细的说明请先翻到后面一章 压缩指令的运用 来查阅，这里鸟哥仅作一个简单的范例而已。 假设我要建立一个空的档案在 /tmp/loopdev ，那可以这样做：

```
[root@linux ~]# dd if=/dev/zero of=/tmp/loopdev bs=1024k count=2048
2048+0 records in
2048+0 records out
# 这个指令在下一小节也会谈到，那个 if 是 input file，
# of 是 output file ，至于 bs 是每个 block 大小，
# count 则是总共几个 bs 的意思。不过，测试时，注意 /tmp
# 那个 partition 的大小啊！
```

格式化

很简单就建立起一个 2GB 的档案了呐！！接下来当然是格式化啰！

```
[root@linux ~]# mke2fs -j /tmp/loopdev
mke2fs 1.35 (28-Feb-2004)
loopdev is not a block special device.
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
262144 inodes, 524288 blocks
26214 blocks (5.00%) reserved for the super user
....以下省略.....
```

挂载

那要如何挂载啊？利用 mount 的特殊参数，那个 -o loop 的参数来处理！

```
[root@linux ~]# mount -t ext3 -o loop /tmp/loopdev /media/cdrom/
[root@linux ~]# df
```

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/tmp/loopdev           2064208     35880   1923472   2% /media/cdrom
```

多了个独立的装置给您使用喔！其实就是那个 2GB 的档案内容啦！ 这东西看起来似乎没有什么用途的样子，不过，如果您未来想要玩 Linux 上面的『虚拟主机』的话， 也就是以一部 Linux 主机再切割成为数个独立的主机系统时，类似 VMware 这类的软件， 在 Linux 上使用 xen 这个软件，他就可以配合这种 loop device 的档案类型来进行根目录的挂载， 真的非常有用的喔！ ^_^

虚拟内存之建置

我们前面谈了很多各式各样的 filesystem ，不过，您晓得在安装的时候设定的那一个『 虚拟内存 (swap) 』要如何增加吗？举个简单的例子吧，鸟哥的 Sun 主机上面，由于跑的程序太庞大了，通常 swap 需要开启到 1GB 左右，但是呢，有的时候还是会不够的！在 Linux 当中，如果您需要使用到很大量的虚拟内存，偏偏当初给的 swap 扇区不够大，那要怎么办呢？ 有什么方法可以来达成：

- 设定一个 swap partition ？
- 建立一个虚拟内存的档案？

怎么说呢？基本上，虚拟内存就是将硬盘规划出一个区间，让内存的数据可以经由硬盘来读取罢了， 那么如果有 swap file 也就够了对不对！是呀！所以这里我们使用两种方法来尝试建立一下 swap 的扩增吧！ 另外， swap 的建立其实也很简单啊！同样的需要先建立出 swap 这个装置或者是档案后， 将他格式化成为 swap 的格式，最后将他挂载到系统上即可！那就来实作看看吧！

建立虚拟内存装置

第一种正规的方法是『直接再加一棵硬盘，并且将其中某个扇区规划为 swap 的 filesystem 』， 呵呵，说的容易，做起来更容易！实际的动作为：

1. 以『 fdisk /dev/hd[a-d] 』先建立一个 partition ，还记得 fdisk 怎么做吗？回去复习一下吧！简单的来说，就是先 (1)建立一个 partition，然后 (2)将该 partition 的 ID 改为 82 这一个 swap 的磁盘档案格式代号就对 啦！这样这一步骤就 OK 啰！

2. 以『 mkswap /dev/hd[a-d][1-16] 』的方式来将您刚刚建置出来的 partition 『格式化为 swap 的档案格式』，很简单吧！这样就格式化 OK 啰！

3. 再来则是将 swap 启动，启动的指令为『 swapon /dev/hd[a-d][1-16] 』，这样就能启动了！很简单吧！这样 swap 就自动加入到内存容量里头去了！

那么如何将 swap 关掉呢？呵呵！很简单呀！就是直接给他 swapoff 就对了！

例题一：如果您的系统是以鸟哥建议的方式来安装的，那么系统应该有一块剩余的空间。请将该剩余的空间格式化成为一个 swap device ，并且挂载到系统上！

建立虚拟内存档案

那么万一我不想新增加一个扇区呢？可不可以使用 swap file 的方式来新增硬盘呀！当然可以啰！ 而且步骤还蛮简单的呢！基本的流程就是：

1. 以 dd 指令来建立 swapfile ；
2. 以 mkswap 来将 swapfile 格式化为 swap 的档案格式；
3. 以 swapon 来启动该档案，使成为 swap ；
4. 以 swapoff 来关闭该档案！

嗯！多说无益！我们来实际的将您的主机系统上面新增 64MB 的虚拟内存吧！如果可能的话， 请您在您的系统上面实际的操作一次底下的步骤，我想，您应该马上会了解实际的操作流程的！ （底下的步骤是可以复原的！！别担心，不过 mkswap 这个指令的下达要小心一点就是了！）

1. 使用 dd 这个指令来新增一个 64MB 的档案在 /tmp 底下：

```
[root@linux ~]# dd if=/dev/zero of=/tmp/swap bs=4k count=16382
16382+0 records in
16382+0 records out
# dd 这个指令是用来转换档案并且 copy 用的；
#    if 指的是要被转换的输入档案格式 /dev/zero 可以由 man zero 来查看内容；
#    of 指的是输出的档案，我们将之输出到 /tmp/swap 这个档案；
#    bs 指的是一个扇区占用几个 kb ；
#    count 指的是要使用多少个 bs ，所以最后的容量为 bs*count = 4k * 16382 ~ 64MB
```

如上所述，我们将建立一个档名为 /tmp/swap 的档案，且其内容共有 64MB 左右大小的档案；

2. 使用 mkswap 将 /tmp/swap 这个档案格式化为 swap 的档案格式：

```
[root@linux ~]# mkswap /tmp/swap
Setting up swapspace version 1, size = 67096576 bytes
# 请注意，这个指令在下达的时候请『特别小心』，因为下错字元控制，
# 将可能使您的 filesystem 挂掉！
```

3. 使用 swapon 来将 /tmp/swap 启动啰！

```
[root@linux ~]# free
             total       used       free     shared    buffers     cached
Mem:         62524      60200       2324          0        716      19492
-/+ buffers/cache:      39992      22532
Swap:       127004       2620     124384
```

```
[root@linux ~]# swapon /tmp/swap
# 不要怀疑！这样就已经将虚拟内存增加 64 MB 啰！如果您需要每次都启动该档案，
那么将 swapon /tmp/swap 写入 /etc/rc.d/rc.local 当中即可！

[root@linux ~]# free
             total       used       free     shared    buffers     cached
Mem:         62524      60240       2284          0        724      19492
-/+ buffers/cache:      40024      22500
Swap:       192524       2620     189904
```

4. 使用 swapoff 关掉 swap file

```
[root@linux ~]# swapoff /tmp/swap
```

没错，就这么简单的将虚拟内存给他新增啰！赶快来去试试看去！ 不过，如果您的 FC4 从来没有经过 update 的话，那么可能会发生一些小困扰， 困扰的原因在此：https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=164937 ， 因为我们尚未学习如何以 rpm 安装软件，所以这里的练习您可以先略过！

虚拟内存的限制

说实话，虚拟内存在目前的桌上型计算机来讲，存在的意义已经不大了！这是因为目前的 x86 主机所含的内存实在都太大了 (一般入门级至少也都有 256MB 了)，所以，我们的 Linux 系统大概都用不到虚拟内存 (swap) 这个玩意儿的。不过， 如果是针对服务器或者是工作站这些常年上线的系统来说的话，那么，无论如何，swap 还是需要建立的。

因为 swap 主要的功能是当物理内存不够时，则某些在内存当中所占的程序会暂时被移动到 swap 当中，让物理内存可以被需要的程序来使用。另外，如果您的主机支持电源管理模式， 也就是说，您的 Linux 主机系统可以进入『休眠』模式的话，那么， 运作当中的程序状态泽会被纪录到 swap 去，以作为『唤醒』主机的状态依据！。 另外，有某些程序在运作时，本来就会利用 swap 的特性来存放一些数据段， 所以，swap 来是需要建立的！只是不需要太大！

不过， swap 在被建立时，是有限制的喔！

- 在核心 2.4.10 版本以后，单一 swap 量已经没有 2GB 的限制了，
- 但是，最多还是仅能建立到 32 个 swap 的数量！
- 而且，由于目前 x86_64 (64 位) 最大内存寻址到 64GB， 因此， swap 总量最大也是仅能达 64GB 就是了！

本章习题练习：
（ 要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看 ）

- 如何增加一颗新的硬盘在你的 Linux 系统当中？请详述流程：

  安装硬盘：关掉 Linux 主机电源，调整 Hard Disk 的 Jump （master 或 slave），串接在 IDE 的接口，请注意，留意你增加的硬盘所串接的 IDE 接口为哪一个插槽，例如你插在 IDE2 的 Master ，则你的硬盘应为 hdc；此外，需要特别留意的是，目前的机器中，如果是 ATA 66 以上的排线（那种很密的排线），那么 master 或者是 slave 在排在线的顺序是固定的！底端的是 Mater 而中间的是 Slave ，这点请稍微注意呦！

  新增硬件于 BIOS：开启计算机后，按 del 键进入 BIOS，选择 IDE Hard Disk Detector 字样的选项， 让 BIOS 去捉硬盘，然后再选择 Save and Exit；不过，较新的机器通常都可以自动侦测了！但是， 如果你的机器是旧型的，那么还是手动来增加硬盘吧！

  Linux 系统侦测：如果你的 Linux 系统有启动 kudzu 这个服务时，那么开机就会自动去侦测新的硬件装置！ Fedora Core IV 预设是有开启这项服务的，除非你关掉他了！OK，假设你有开启这项服务，那么开机进入 Linux 的时候，系统会告诉你有捉到一个新的硬件，你可以按『configure』由系统直接安装即可；

  格式化硬盘：以 root 的身份进入 Linux 后，执行以下两个程序：fdisk /dev/hd[a-d] 与 mke2fs /dev/hd[a-d][1-16] 。

  建立 mount point：假设我的这颗硬盘要挂在 /disk3 下面，那么就需要 ： mkdir /disk3

  开机自动加载(mount)：再来则是以 vi 修改 /etc/fstab 档案，让每次开机把这个硬盘直接挂入系统中。

  安装完成：你可以使用 mount -a 来将全部的装置重新挂载一遍，或者是重新开机就可以啦！

- 假设条件：我原先规划的 /home 只有 1GB ，但是目前的使用者日众，所以容量不足！我想要增加一棵 8GB 的旧硬盘，要如何作？！

  将硬盘加入 Linux 系统中：利用刚刚上一题的方式将你的硬盘加入到 Linux 系统中，亦即是使用 fdisk 与 mke2fs 建立了 ext2 的档案格式的硬盘!好了,假设该硬盘的代号为 /dev/hdc1 好了！

  挂载新硬盘：由于我需要将新旧扇区都挂上来，这样才有办法将数据由旧硬盘移到新硬盘上面，OK！ 我就建立一个暂存的目录，称为 /disk-tmp：

  ```
  mkdir /disk-tmp
  mount -t ext2 /dev/hdc1 /disk-tmp
  ```

如此一来则 /disk-tmp 就是新挂上来那颗 8 GB 的硬盘啦！

移动数据：好了！现在开始将数据 copy 到新挂上的硬盘上面吧！

```
cd /home
tar -zcvf /disk-tmp/home.tar.gz *
cd /disk-tmp
tar -zxvf home.tar.gz
```

上面的指令会将目前旧有的 /home 底下的东西完全的压缩之后移动到 /disk-tmp/home.tar.gz 这个压缩档，然后再到 /disk-tmp 底下将他解压缩！这样数据就复制到新挂上来的硬盘啦！ 卸载旧的，挂上新的：好了，那么我们就开始来测试一下吧！你可以这样做：

```
umount /home
mount -t ext2 /dev/hdc1 /home
```

注意呦！如果你的 /home 底下原本就没有挂载扇区的话，那么你就可以直接将 /home 底下的数据都砍掉，然后在挂上新的那颗硬盘就好了！而 home.tar.gz 这个档案就可以用作为备份之用！

开机执行：同样的，如果要设定成开机就挂上这颗新的硬盘，那就修改 /etc/fstab 档案吧！

- 如果扇区 /dev/hda3 有问题，偏偏他是被挂载上的，请问我要如何修理此一扇区？

```
umount /dev/hda3
fsck /dev/hda3
```

- 我们常常说，开机的时候，『发现硬盘有问题』，请问，这个问题的产生是『filesystem 的损毁』，还是『硬盘的损毁』？

  特别需要注意的是，如果您某个 filesystem 里面，由于操作不当，可能会造成 Superblock 数据的损毁， 或者是 inode 的架构损毁，或者是 block area 的记录遗失等等，这些问题当中，其实您的『硬盘』还是好好的， 不过，在硬盘上面的『档案系统』则已经无法再利用！一般来说，我们的 Linux 很少会造成 filesystem 的损毁， 所以，发生问题时，很可能整个硬盘都损毁了。但是，如果您的主机常常不正常断电，那么， 很可能硬盘是没问题的，但是，档案系统则有损毁之虞。此时，重建档案系统 (reinstall) 即可！ 不需要换掉硬盘啦！ ^_^

- 当我有两个档案，分别是 file1 与 file2 ，这两个档案互为 hard link 的档案，请问， 若我将 file1 删除，然后再以类似 vi 的方式重新建立一个名为 file1 的档案， 则 file2 的内容是否会被更动？

  这是来自网友的疑问。当我删除 file1 之后， file2 则为一个正规档案，并不会与他人共同分享同一个 inode 与 block ，因此，当我重新建立一个档名为 file1 时，他所利用的 inode 与 block 都是由我们的 filesystem 主动去搜寻 meta data ，找到空的 inode 与 block 来建立的， 与原本的 file1 并没有任何关连性喔！所以，新建的 file1 并不会影响 file2 呢！

参考数据：

- 硬盘的相关认识：http://www.linwei.com.tw/knowhdd.html
- Linux System Administrator's Survival Guide
  http://sunsite.iisc.ernet.in/virlib/linux/survival/ewtoc.html
- Design and Implementation of the Second Extended Filesystem
  http://e2fsprogs.sourceforge.net/ext2intro.html
- 小木偶的汇编语言教学之硬盘知识：
  http://home.educities.edu.tw/wanker742126/asm/ch32.html
- Linux 核心所支持的装置代号查询：
  http://www.kernel.org/pub/linux/docs/device-list/devices.txt

Linux 档案与目录管理

最近更新日期：2005/08/22

在前一章节里面我们认识了 Linux 系统下的档案权限概念以及目录的配置说明。 在这个章节当中，我们就直接来进一步的操作与管理档案与目录吧！包括在不同的目录间变换、 建立与删除目录、建立与删除档案，还有寻找档案、查阅档案内容等等， 都会在这个章节作个简单的介绍啊！

目录与路径：

由前一章节『Linux 的档案权限与目录配置』中约略了解到 Linux 的『树状目录』概念之后，接下来就得要实际的来搞定一些基本的路径问题了！这些目录的问题当中， 最重要的莫过于『绝对路径』与『相对路径』的意义啦！赶紧来了解一下！

相对路径与绝对路径：

在开始目录的切换之前，你必须要先了解一下所谓的『路径 (PATH)』， 有趣的是：什么是『相对路径』与『绝对路径』？ 虽然前一章已经稍微针对这个议题提过一次，不过，这里不厌其烦的再次的强调一下！

如果你还记得前一章的内容的话，那么应该还记得 Linux 里面的目录是呈现『树状目录』 的情况，就是有分支的啦！好了，假设你需要在任意一个目录下变换到根目录的 etc 底下，那么你就应该要使用『 cd

/etc 』这个情况， 这也就是所谓的『绝对路径』，他是从根目录连续写上来的一个情况， 所以不论你在哪一个路径现执行这一个指令，都会将你移动到该路径下。那如果我是使用 『cd etc 』呢？那表示你要切换到『目前这个目录下的 etc 目录中』，情况可是不一样的呦！通常第一次接触 Linux 的使用者常会搞错这一个路径的观念！

- 绝对路径：路径的写法『一定由根目录 / 写起』，例如： /usr/share/doc 这个目录。
- 相对路径：路径的写法『不是由 / 写起』，例如由 /usr/share/doc 要到 /usr/share/man 底下时，可以写成： 『cd ../man』这就是相对路径的写法啦！相对路径意指『相对于目前工作目录的路径！』

那么相对路径与绝对路径有什么了不起呀！？喝！那可真的是了不起了！假设您写了一个套件， 这个套件共需要三个目录，分别是 etc, bin, man 这三个目录，然而由于不同的人喜欢安装在不同的目录之下， 假设甲安装的目录是 /usr/local/packages/etc, /usr/local/packages/bin 及 /usr/local/packages/man ，不过乙却喜欢安装在 /home/packages/etc, /home/packages/bin, /home/packages/man 这三个目录中，请问如果需要用到绝对路径的话，那么是否很麻烦呢？是的！ 如此一来每个目录下的东西就很难对应的起来！这个时候相对路径的写法就显的特别的重要了！

此外，如果您跟鸟哥一样，喜欢将路径的名字写的很长，好让自己知道那个目录是在干什么的，例如：/data4/staiwan19961109/models-3/smoke 这个目录，而另一个目录在 /data4/staiwan19961109/models-3/cctm ，那么我从第一个要到第二个目录去的话，怎么写比较方便？ 当然是『 cd ../cctm 』比较方便啰！对吧！

但是对于档案的正确性来说，『绝对路径的正确度要比较好～』。 一般来说，鸟哥会建议您，如果是在写程序 (shell scripts) 的条件下，务必使用绝对路径的写法。 怎么说呢？因为绝对路径的写法虽然比较麻烦，但是可以肯定这个写法绝对不会有问题。 如果使用相对路径在程序当中，则可能由于您执行的工作环境不同，导致一些问题的发生。 这个问题在例行性命令当中尤其重要！这个现象我们在 shell script 时，会再次的提醒您喔！ ^_^

目录的相关操作：

在之前我们稍微提到这个变换目录的指令是 cd ，还有哪些可以进行目录操作的指令呢？ 例如建立目录啊、删除目录之类的～还有，得要先知道的，就是有哪些比较特殊的目录呢？ 举例来说，底下这些就是比较特殊的目录，得要用力的记下来才行：

```
.         代表此层目录
..        代表上一层目录
-         代表前一个工作目录
~         代表『目前使用者身份』所在的家目录
~account  代表 account 这个使用者的家目录
```

而在目录底下有两个目录是一定会存在的！那就是 . 与 .. 啰～ 分别代表此层与上层目录的意思。那我们在前一章 Linux 档案属性与目录配置 里面也知道根目录 (/) 是所有目录的最顶层，那么 / 有 .. 吗？！您可以使用 ls -al / 去看看， 答案是『有的！』不过，您也可以查阅到，根目录的 . 与 .. 属性完全一模一样，哈哈！ 原来根目录的顶层 (..) 与他自己 (.) 是同一个目录啦！ ^_^

底下我们就来谈一谈几个常见的处理目录的指令吧：

- cd：变换目录
- pwd：显示目前的目录
- mkdir：建立一个新的目录
- rmdir：删除一个空的目录

- cd (变换目录)

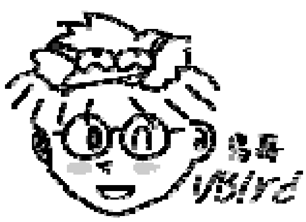
我们知道 dmtsai 这个使用者的家目录是 /home/dmtsai ，而 root 家目录则是 /root ，假设我以 root 身份在 Linux 系统中，那么简单的说明一下这几个特殊的目录的意义是：

```
[root@linux ~]# cd [相对路径或绝对路径]
# 最重要的就是目录的绝对路径与相对路径，还有一些特殊目录的符号啰！
[root@linux ~]# cd ~dmtsai
# 代表去到 dmtsai 这个使用者的家目录，亦即 /home/dmtsai
[root@linux dmtsai]# cd ~
# 表示回到自己的家目录，亦即是 /root 这个目录
[root@linux ~]# cd
# 没有加上任何路径，也还是代表回到自己家目录的意思喔！
[root@linux ~]# cd ..
# 表示去到目前的上层目录，亦即是 /root 的上层目录的意思；
[root@linux /]# cd -
# 表示回到刚刚的那个目录，也就是 /root 啰～
[root@linux ~]# cd /var/spool/mail
# 这个就是绝对路径的写法！直接指定要去的完整路径名称！
[root@linux mail]# cd ../mqueue
# 这个是相对路径的写法，我们由 /var/spool/mail 去到 /var/spool/mqueue 就这样写！
```

cd 是 Change Directory 的缩写，这是用来变换工作目录的指令。注意，目录名称与 cd 指令之间存在一个空格。 一登入 Linux 系统后，root 会在 root 的家目录，亦即 /root 下，OK！那回到上一层目录可以用『 cd .. 』。 利用相对路径的写法必须要确认您目前的路径才能正确的去到想要去的目录。例如上表当中最后一个例子， 您必须要确认您是在 /var/spool/mail 当中，并且知道在 /var/spool 当中有个 mqueue 的目录才行啊～ 这样才能使用 cd ../mqueue 去到正确的目录说，否则就要直接输入 cd /var/spool/mqueue 啰～

其实，我们的提示字符，亦即那个 [root@linux ~]# 当中，就已经有指出目前的目录了， 刚登入时会到自己的家目录，而家目录还有一个代码，那就是『 ~ 』符号！ 例如上面的例子可以发现，使用『 cd ~ 』可以回到个人的家目录里头去呢！ 另外，针对 cd 的使用方法，如果仅输入 cd 时，代表的就是『 cd ~ 』的意思喔～ 亦即是会回到自己的家目录啦！而那个『 cd - 』比较难以理解，请自行多做几次练习， 就会比较明白了。

> Tips:
> 还是要一再地提醒，我们的 Linux 的预设指令列模式 (bash shell) 具有档案补齐功能， 您要常常利用 [tab] 按键来达成您的目录完整性啊！这可是个好习惯啊～ 可以避免您按错键盘输入错字说～ ^_^



---

* pwd (显示目前所在的目录)

```
[root@linux ~]# pwd [-P]
参数:
-P  : 显示出确实的路径，而非使用连结 (link) 路径。
范例:
[root@linux ~]# pwd
/root   <== 显示出目录啦~
[root@linux ~]# cd /var/mail
[root@linux mail]# pwd
/var/mail
[root@linux mail]# pwd -P
/var/spool/mail   <== 怎么回事? 有没有加 -P 差很多~
[root@linux mail]# ls -l /var/mail
lrwxrwxrwx  1 root root 10 Jun 25 08:25 /var/mail -> spool/mail
# 看到这里应该知道为啥了吧? 因为 /var/mail 是连结档，连结到 /var/spool/mail
# 所以，加上 pwd -P 的参数后，会不以连结文件的数据显示，而是显示正确的完整路径啊!
```

pwd 是 Print Working Directory 的缩写，也就是显示目前所在目录的指令， 例如在上个表格最后的目录是 /var/mail 这个目录，但是提示字符仅显示 mail， 如果你想要知道目前所在的目录，可以输入 pwd 即可。此外，由于很多的套件所使用的目录名称都相同，例如 /usr/local/etc 还有 /etc ，但是通常 Linux 仅列出最后面那一个目录而已，这个时候你就可以使用 pwd 来知道你的所在目录啰! 免得搞错目录，结果……

其实有趣的是那个 -P 的参数啦! 他可以让我们取得正确的目录名称，而不是以连结文件的路径来显示的。如果您是 Fedora Core 4 的话，刚刚好， /var/mail 是 /var/spool/mail 的连结档， 所以，透过到 /var/mail 下达 pwd -P 就能够知道这个参数的意义啰~ ^_^

---

* mkdir (建立新目录)

```
[root@linux ~]# mkdir [-mp] 目录名称
参数:
-m : 设定档案的权限喔! 直接设定，不需要看预设权限 (umask) 的脸色~
-p : 帮助你直接将所需要的目录递归建立起来!
范例:
[root@linux ~]# cd /tmp
[root@linux tmp]# mkdir test    <== 建立一名为 test 的新目录
[root@linux tmp]# mkdir test1/test2/test3/test4
mkdir: cannot create directory `test1/test2/test3/test4':
No such file or directory  <== 没办法直接建立此目录啊!
[root@linux tmp]# mkdir -p test1/test2/test3/test4
# 加了这个 -p 的参数，可以自行帮您建立多层目录!
[root@linux tmp]# mkdir -m 711 test2
[root@linux tmp]# ls -l
drwxr-xr-x  3 root  root 4096 Jul 18 12:50 test
```

```
drwxr-xr-x  3 root  root 4096 Jul 18 12:53 test1
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
# 仔细看上面的权限部分，如果没有加上 -m 来强制设定属性，系统会使用预设属性。
# 那么您的预设属性为何？这要透过底下介绍的 umask 才能了解喔！ ^_^
```

如果想要建立新的目录的话，那么就使用 mkdir (make directory) 吧！ 不过，请注意呦！在预设的情况下，你所需要的目录得一层一层的建立才行!例如:假如你要建立一个目录为 /home/bird/testing/test1，那么首先必须要有 /home 然后 /home/bird ，再来 /home/bird/testing 都必须要存在，才可以建立 /home/bird/testing/test1 这个目录！假如没有 /home/bird/testing 时，就没有办法建立 test1 的目录啰！不过，现在有个更简单有效的方法啦！那就是加上 -p 这个参数喔！你可以直接下达：『 mkdir -p /home/bird/testing/test1 』 则系统会自动的帮你将 /home, /home/bird, /home/bird/testing 依序的建立起目录！并且， 如果该目录本来就已经存在时，系统也不会显示错误讯息喔！挺快乐的吧！ ^_^

另外，有个地方您必须要先有概念，那就是『预设权限』的地方。我们可以利用 -m 来强制给予一个新的目录相关的属性， 例如上表当中，我们给予 -m 711 来给予新的目录 drwx--x--x 的属性。不过，如果没有给予 -m 属性时， 那么预设的新建目录属性又是什么呢？这个跟 umask 有关，我们在后头会加以介绍的。

- rmdir (删除『空』的目录)

```
[root@linux ~]# rmdir [-p] 目录名称
参数：
-p ：连同上层『空的』目录也一起删除
范例：
[root@linux tmp]# ls -l
drwxr-xr-x  3 root  root 4096 Jul 18 12:50 test
drwxr-xr-x  3 root  root 4096 Jul 18 12:53 test1
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
[root@linux tmp]# rmdir test
[root@linux tmp]# rmdir test1
rmdir: `test1': Directory not empty
[root@linux tmp]# rmdir -p test1/test2/test3/test4
[root@linux tmp]# ls -l
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
# 瞧！利用 -p 这个参数，立刻就可以将 test1/test2/test3/test4 一次删除～
# 不过要注意的是，这个 rmdir 仅能『删除空的目录』喔！
```

如果想要建立删除旧有的目录时，就使用 rmdir 吧！例如将刚刚建立的 test 杀掉，使用 rmdir test 即可！请注意呦！目录需要一层一层的删除才行！而且 被删除的目录里面必定不能还有其它的目录或档案！这也是所谓的空的目录 (empty directory) 的意思啊！那如果要将所有目录下的东西都杀掉呢？！ 这个时候就必须使用 rm -rf test 啰！不过，还是使用 rmdir 比较不危险！不过，你也可以尝试以 -p 的参数加入，来删除上层的目录喔！

关于执行文件路径的变量： $PATH

在提过了绝对路径、相对路径与指令的下达方式之后，您应该会稍微注意到一件事情，那就是：『 为什么我可以在任何地方执行 /bin/ls 这个指令呢？ 』对呀！ 为什么我可以直接执行 ls 就一定可以显示出一些讯息而不会说找不到该 /bin/ls 指令呢？ 这是因为环境变量 PATH 的帮助所致呀！当我们在执行一个指令的时候， 系统会依照 PATH 的设定去每个 PATH 定义的路径下搜寻执行文件，先搜寻到的指令先被执行之！现在，请下达 echo $PATH ， echo 有『显示、印出』的意思，而 PATH 前面加的 $ 表示后面接的是变量，所以即会显示出目前的 PATH 了！

```
[root@linux ~]# echo $PATH
/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:/usr/local/sbin
```

注意到了吗？对啦！ /bin 在 PATH 的设定之中，所以自然就可以找的到 ls 啦！ PATH 对于执行档来说，是个很重要的『变量』，他主要是用来规范指令搜寻的目录。 而每个目录是有顺序的，每个目录中间以冒号『:』来分隔，就如同上面范例中提到的啰！ 那么 PATH 这个变量还有什么地方重要呢？

- 如果你将 ls 移动到 /root 底下的话 (mv /bin/ls /root) ，然后你自己本身也在 /root 底下 (cd /root) ，但是当你执行 ls 的时候，他就是不理你？怎么办？这是因为 PATH 这个变量没有 /root 这个目录，而你又将 ls 移动到 /root 底下了，自然系统就找不到可执行文件了，因此就会告诉你， command not found ！那么该怎么克服这种问题呢？有两个方法，其一：直接将 /root 的路径加入 PATH 当中！如何增加？可以使用：

  ```
  [root@linux ~]# PATH="$PATH":/root
  ```

  这种方式来增加 PATH 搜寻目录即可！另一种方式则是使用完整文件名来下达指令， 亦即直接使用相对或绝对路径来执行，例如：

  ```
  [root@linux ~]# /root/ls
  [root@linux ~]# ./ls
  ```

  因为在同一个目录中，而我们又知道在同一个目录中的目录符号为『 . 』，因此，就以上面的 ./ls 来执行也可以！这种执行方式以后您应该会很常见到才对！

- 如果我有两个 ls 档案在不同的目录中，例如 /usr/local/bin/ls 底下与 /bin/ls 那么当我下达 ls 的时候，哪个 ls 会被执行？那还用说，就找出 PATH 里面哪个目录先被查询，则那个目录下的档案就会被先执行了！

- 咦！既然如此的话，那么为何不要在 PATH 里面加入 . 这个目录，如此一来的话， 不就可以直接在所在目录执行档案了吗？因为 . 代表所在目录嘛！是这样没错！但是有没有想过，如果某天，某个怪怪的使用者在 /tmp 里面写了一个 ls 的档案，偏偏他是有害的档案，那么当你在 /tmp 底下执行 ls 时，怎么办？！没错，可能会『中标』，所以啰，为了安全起见，不建议将『 . 』加入 PATH 的搜寻当中！

关于更多的 PATH 与相关的『变量』及『环境变量』概念，我们会在第三篇 bash shell 时， 再更深入的介绍啰～而经过上面的说明，您应该也能够比较了解的是：『 为什么绝对路径下达指令的方法比相对路径要正确的多 』这句话的意义啰～ ^_^ 因为是直接找到该指令来执行，而不是透过 PATH 这个变量的内容去搜寻的啊！

档案与目录管理：

谈了谈目录与路径之后，再来讨论一下关于档案的一些基本管理吧！档案与目录的管理上，不外乎『显示属性』、 『拷贝』、『删除档案』及『移动档案或目录』等等，由于档案与目录的管理在 Linux 当中是很重要的！ 尤其是每个人自己家目录的数据也都需要注意管理！由于我们在执行程序的时后，系统预设有一个搜寻的路径顺序， 如果有两个以上相同档名的执行档分别在不同的路径时，呵呵，就需要特别留意啰！这里我们来谈一谈有关档案与目录的一些基础管理部分吧！

档案与目录的检视： ls

```
[root@linux ~]# ls [-aAdfFhilRS] 目录名称
[root@linux ~]# ls [--color={none,auto,always}] 目录名称
[root@linux ~]# ls [--full-time] 目录名称
参数：
-a  ：全部的档案，连同隐藏档( 开头为 . 的档案) 一起列出来～
-A  ：全部的档案，连同隐藏档，但不包括 . 与 .. 这两个目录，一起列出来～
-d  ：仅列出目录本身，而不是列出目录内的档案数据
-f  ：直接列出结果，而不进行排序 (ls 预设会以档名排序！)
-F  ：根据档案、目录等信息，给予附加数据结构，例如：
      *：代表可执行档； /：代表目录； =：代表 socket 档案； |：代表 FIFO 档案；
-h  ：将档案容量以人类较易读的方式(例如 GB, KB 等等)列出来；
-i  ：列出 inode 位置，而非列出档案属性；
-l  ：长数据串行出，包含档案的属性等等数据；
-n  ：列出 UID 与 GID 而非使用者与群组的名称 (UID 与 GID 会在账号管理提到！)
-r  ：将排序结果反向输出，例如：原本档名由小到大，反向则为由大到小；
-R  ：连同子目录内容一起列出来；
-S  ：以档案容量大小排序！
-t  ：依时间排序
--color=never  ：不要依据档案特性给予颜色显示；
--color=always ：显示颜色
--color=auto   ：让系统自行依据设定来判断是否给予颜色
--full-time    ：以完整时间模式 (包含年、月、日、时、分) 输出
--time={atime,ctime} ：输出 access 时间或 改变权限属性时间 (ctime)
                       而非内容变更时间 (modification time)
范例：
```

在 Linux 系统当中，这个 ls 指令可能是最常被执行的吧！因为我们随时都要知道档案或者是目录的相关信息啊～ 不过，我们 Linux 的档案所记录的信息实在是太多了， ls 没有需要全部都列出来呢～ 所以，当您只有下达 ls 时，预设显示的只有：非隐藏档的档名、 以档名进行排序及文件名代表的颜色显示；如此而已。举例来说， 您下达 ls /etc 之后，只有经过排序的文件名以及以蓝色显示目录及白色显示一般档案，如此而已。

那如果我还想要加入其它的显示信息时，可以加入上头提到的那些有用的参数呢～ 举例来说，我们之前一直用到的 -l 这个长串显示数据内容，以及将隐藏档也一起列示出来的 -a 参数等等。

```
范例一：将家目录下的所有档案列出来(含属性与隐藏文件)
[root@linux ~]# ls -al ~
total 252
drwxr-x---   9 root root  4096 Jul 16 23:40 .
drwxr-xr-x  24 root root  4096 Jul 16 23:45 ..
-rw-------   1 root root  1491 Jun 25 08:53 anaconda-ks.cfg
-rw-------   1 root root 12543 Jul 18 01:23 .bash_history
-rw-r--r--   1 root root    24 Dec  4  2004 .bash_logout
-rw-r--r--   1 root root   191 Dec  4  2004 .bash_profile
-rw-r--r--   1 root root   395 Jul  4 11:45 .bashrc
-rw-r--r--   1 root root 68495 Jun 25 08:53 install.log
-rw-r--r--   1 root root  5976 Jun 25 08:53 install.log.syslog
drwx------   2 root root  4096 Jul  4 16:03 .ssh
-rw-------   1 root root 12613 Jul 16 23:40 .viminfo
# 这个时候您会看到以 . 为开头的几个档案，以及目录文件 ./../.ssh 等等，
# 不过，目录文件都是以深蓝色显示，有点不容易看清楚就是了。

范例二：承上题，不显示颜色，但在文件名末显示出该文件名代表的类型(type)
[root@linux ~]# ls -alF --color=never  ~
total 252
drwxr-x---   9 root root  4096 Jul 16 23:40 ./
drwxr-xr-x  24 root root  4096 Jul 16 23:45 ../
-rw-------   1 root root  1491 Jun 25 08:53 anaconda-ks.cfg
-rw-------   1 root root 12543 Jul 18 01:23 .bash_history
-rw-r--r--   1 root root    24 Dec  4  2004 .bash_logout
-rw-r--r--   1 root root   191 Dec  4  2004 .bash_profile
-rw-r--r--   1 root root   395 Jul  4 11:45 .bashrc
-rw-r--r--   1 root root 68495 Jun 25 08:53 install.log
-rw-r--r--   1 root root  5976 Jun 25 08:53 install.log.syslog
drwx------   2 root root  4096 Jul  4 16:03 .ssh/
-rw-------   1 root root 12613 Jul 16 23:40 .viminfo
# 注意看到显示结果的第一行，嘿嘿～知道为何我们会下达类似 ./command
# 之类的指令了吧？因为 ./ 代表的是『目前目录下』的意思啊！至于什么是 FIFO/Socket ？
# 请参考前一章节的介绍啊！

范例三：完整的呈现档案的修改时间 *(modification time)
[root@linux ~]# ls -al --full-time  ~
total 252
drwxr-x---   9 root root  4096 2005-07-16 23:40:13.000000000 +0800 .
drwxr-xr-x  24 root root  4096 2005-07-16 23:45:05.000000000 +0800 ..
-rw-------   1 root root  1491 2005-06-25 08:53:37.000000000 +0800 anaconda-ks.cfg
```

```
-rw-------    1 root root 12543 2005-07-18 01:23:33.000000000 +0800 .bash_history
-rw-r--r--    1 root root    24 2004-12-04 05:44:13.000000000 +0800 .bash_logout
-rw-r--r--    1 root root   191 2004-12-04 05:44:13.000000000 +0800 .bash_profile
-rw-r--r--    1 root root   395 2005-07-04 11:45:16.000000000 +0800 .bashrc
-rw-r--r--    1 root root 68495 2005-06-25 08:53:34.000000000 +0800 install.log
-rw-r--r--    1 root root  5976 2005-06-25 08:53:28.000000000 +0800 install.log.syslog
drwx------    2 root root  4096 2005-07-04 16:03:24.000000000 +0800 .ssh
-rw-------    1 root root 12613 2005-07-16 23:40:13.000000000 +0800 .viminfo
# 请仔细看，上面的『时间』字段变了喔！变成较为完整的格式。
# 一般来说， ls -al 仅列出目前短格式的时间，有时不会列出年份，
# 藉由 --full-time 可以查阅到比较正确的完整时间格式啊！
```

其实 ls 的用法还有很多，包括查阅档案所在 i-node 的 ls -i 参数，以及用来进行档案排序的 -S 参数，还有用来查阅不同时间的动作的 --time=atime 等参数。而这些参数的存在都是因为 Linux 档案系统记录了很多有用的信息的缘故。那么 Linux 的档案系统中，这些与权限、属性有关的数据放在哪里呢？ 放在 i-node 里面。关于这部分，我们会在下个章节继续跟您作比较深入的介绍啊！

无论如何， ls 最常被使用到的功能还是那个 -l 的参数，为此，很多 distribution 在预设的情况中， 已经将 ll (L 的小写) 设定成为 ls -l 的意思了！其实，那个功能是 Bash shell 的 alias 功能呢～ 也就是说，我们直接输入 ll 就等于是输入 ls -l 是一样的～关于这部分，我们会在第三章 bash shell 时再次的强调滴～

复制、移动与删除： cp, rm, mv

要复制档案，请使用 cp (copy) 这个指令即可～不过， cp 这个指令的用途可多了～ 除了单纯的复制之外，还可以建立连结档 (就是快捷方式啰)，比对两档案的新旧而予以更新， 以及复制整个目录等等的功能呢！至于移动目录与档案，则使用 mv (move)， 这个指令也可以直接拿来作更名 (rename) 的动作喔！至于移除吗？那就是 rm (remove) 这个指令啰～底下我们就来瞧一瞧先～

- cp (复制档案或目录)

```
[root@linux ~]# cp [-adfilprsu] 来源档(source) 目的檔(destination)
[root@linux ~]# cp [options] source1 source2 source3 .... directory
参数：
-a  : 相当于 -pdr 的意思；
-d  : 若来源文件为连结文件的属性(link file)，则复制连结文件属性而非档案本身；
-f  : 为强制 (force) 的意思，若有重复或其它疑问时，不会询问使用者，而强制复制；
-i  : 若目的檔(destination)已经存在时，在覆盖时会先询问是否真的动作！
-l  : 进行硬式连结 (hard link) 的连结档建立，而非复制档案本身；
-p  : 连同档案的属性一起复制过去，而非使用预设属性；
-r  : 递归持续复制，用于目录的复制行为；
-s  : 复制成为符号连结文件 (symbolic link)，亦即『快捷方式』档案；
-u  : 若 destination 比 source 旧才更新 destination ！
```

```
最后需要注意的，如果来源档有两个以上，则最后一个目的文件一定要是『目录』才行！
范例：
范例一：将家目录下的 .bashrc 复制到 /tmp 下，并更名为 bashrc
[root@linux ~]# cd /tmp
[root@linux tmp]# cp ~/.bashrc bashrc
[root@linux tmp]# cp -i ~/.bashrc bashrc
cp: overwrite `basrhc'? n
# 重复作两次动作，由于 /tmp 底下已经存在 bashrc 了，加上 -i 参数，
# 则在覆盖前会询问使用者是否确定！可以按下 n 或者 y 呢！
# 但是，反过来说，如果不想要询问时，则加上 -f 这个参数来强制直接覆盖！

范例二：将 /var/log/wtmp 复制到 /tmp 底下
[root@linux tmp]# cp /var/log/wtmp . <==想要复制到目前的目录，最后的 . 不要忘
[root@linux tmp]# ls -l /var/log/wtmp wtmp
-rw-rw-r--  1 root utmp 71808 Jul 18 12:46 /var/log/wtmp
-rw-r--r--  1 root root 71808 Jul 18 21:58 wtmp
# 注意到了吗？！在不加任何参数的情况下，档案的所属者会改变，连权限也跟着改变了～
# 这是个很重要的特性！要注意喔！还有，连档案建立的时间也不一样了！
# 如果您想要将档案的所有特性都一起复制过来，可以加上 -a 喔！
[root@linux tmp]# cp -a /var/log/wtmp wtmp_2
[root@linux tmp]# ls -l /var/log/wtmp wtmp_2
-rw-rw-r--  1 root utmp 71808 Jul 18 12:46 /var/log/wtmp
-rw-rw-r--  1 root utmp 71808 Jul 18 12:46 wtmp_2
# 瞭了吧！整个资料特性完全一模一样ㄟ！真是不赖～这就是 -a 的特性！

范例三：复制 /etc/ 这个目录下的所有内容到 /tmp 底下
[root@linux tmp]# cp /etc/ /tmp
cp: omitting directory `/etc'   <== 如果是目录，不能直接复制，要加上 -r 的参数
[root@linux tmp]# cp -r /etc/ /tmp
# 还是要再次的强调喔！ -r 是可以复制目录，但是，档案与目录的权限会被改变～
# 所以，也可以利用 cp -a /etc /tmp 来下达指令喔！

范例四：将范例一复制的 bashrc 建立一个连结档 (symbolic link)
[root@linux tmp]# ls -l bashrc
-rw-r--r--  1 root root 395 Jul 18 22:08 bashrc
[root@linux tmp]# cp -s bashrc bashrc_slink
[root@linux tmp]# cp -l bashrc bashrc_hlink
[root@linux tmp]# ls -l bashrc*
-rw-r--r--  2 root root 395 Jul 18 22:08 bashrc
-rw-r--r--  2 root root 395 Jul 18 22:08 bashrc_hlink
lrwxrwxrwx  1 root root   6 Jul 18 22:31 bashrc_slink -> bashrc
# 那个 bashrc_slink 是由 -s 的参数造成的，建立的是一个『快捷方式』，
# 所以您会看到在档案的最右边，会显示这个档案是『连结』到哪里去的！
```

```
# 至于那个 bashrc_hlink 有趣了！建立了这个档案之后， bashrc 与 bashrc_hlink
# 所有的参数都一样，只是，第二栏的 link 数改变成为 2 了～而不是原本的 1 喔！
# 这两种连结的方式的异同，我们会在下一章里面进行介绍的！

范例五：若 ~/.bashrc 比 /tmp/bashrc 新才复制过来
[root@linux tmp]# cp -u ~/.bashrc /tmp/bashrc
# 这个 -u 的特性，是在目标档案与来源档案有差异时，才会复制的。
# 所以，比较常被用于『备份』的工作当中喔！ ^_^

范例六：将范例四造成的 bashrc_slink 复制成为 bashrc_slink_2
[root@linux tmp]# cp bashrc_slink bashrc_slink_2
[root@linux tmp]# ls -l bashrc_slink*
lrwxrwxrwx  1 root root   6 Jul 18 22:31 bashrc_slink -> bashrc
-rw-r--r--  1 root root 395 Jul 18 22:48 bashrc_slink_2
# 这个例子也是很有趣喔！原本复制的是连结档，但是却将连结档的实际档案复制过来了
# 也就是说，如果没有加上任何参数时，复制的是源文件，而非连结文件的属性！
# 若要复制连结文件的属性，就得要使用 -d 或者 -a 的参数了！

范例七：将家目录的 .bashrc 及 .bash_history 复制到 /tmp 底下
[root@linux tmp]# cp ~/.bashrc ~/.bash_history /tmp
# 可以将多个数据一次复制到同一个目录去！
```

这个 cp 的功能很多，而由于我们常常在进行一些数据的复制，所以也会常常用到这个指令的。 一般来说，我们如果去复制别人的数据 (当然，该档案您必须要有 read 的权限才行啊！ ^_^) 时， 总是希望复制到的数据最后是我们自己的，所以，在预设的条件中， cp 的来源档与目的档的权限是不同的，目的档的拥有者通常会是指令操作者本身。举例来说， 上面的范例二中，由于我是 root 的身份，因此复制过来的档案拥有者与群组就改变成为 root 所有了！ 这样说，可以明白吗？！ ^_^

由于具有这个特性，因此，当我们在进行备份的时候，某些需要特别注意的特殊权限档案， 例如密码文件 (/etc/shadow) 以及一些设定档，就不能直接以 cp 来复制，而必须要加上 -a 或者是 -p 等等可以完整复制档案权限的参数才行！另外，如果您想要复制档案给其它的使用者， 也必须要注意到档案的权限(包含读、写、执行以及档案拥有者等等)， 否则，其它人还是无法针对您给予的档案进行修订的动作喔！注意注意！

至于上面的范例当中，第四个范例是最有趣的，使用 -l 及 -s 都会建立所谓的连结档 (link file)， 但是这两种连结档确有不一样的展现情况。这是怎么一回事啊？ 那个 -l 就是所谓的 hard link ，至于 -s 则是 symbolic link ，鸟哥这里先不介绍， 因为这个涉及 i-node 的相关知识，我们还没有介绍到，下一章再来讨论这个 link 的问题喔！ 总之，由于 cp 有种种的档案属性与权限的特性，所以，在复制时，您必须要清楚的了解到：

- 是否需要完整的保留来源档案的信息？
- 来源档案是否为连结档 (symbolic link file)？
- 来源档是否为特殊的档案，例如 FIFO, socket 等？
- 来源文件是否为目录？

* rm（移除档案或目录）

```
[root@linux ~]# rm [-fir] 档案或目录
参数：
-f  ：就是 force 的意思，强制移除；
-i  ：互动模式，在删除前会询问使用者是否动作
-r  ：递归删除啊！最常用在目录的删除了
范例：
范例一：建立一档案后予以删除
[root@linux ~]# cd /tmp
[root@linux tmp]# cp ~/.bashrc bashrc
[root@linux tmp]# rm -i bashrc
rm: remove regular file `bashrc'? y
# 如果加上 -i 的参数就会主动询问喔！那么如果不要询问呢？就加 -f 参数啊！

范例二：删除一个不为空的目录
[root@linux tmp]# mkdir test
[root@linux tmp]# cp ~/.bashrc test/ <== 将档案复制到此目录去，就不是空的目录了
[root@linux tmp]# rmdir test
rmdir: `test': Directory not empty <== 删不掉啊！因为这不是空的目录！
[root@linux tmp]# rm -rf test

范例三：删除一个带有 - 开头的档案
[root@linux tmp]# ls *aa*
-rw-r--r--  1 root  root      0 Aug 22 10:52 -aaa-
[root@linux tmp]# rm -aaa-
rm: invalid option -- a
Try `rm --help' for more information.  <== 因为 "-" 是参数嘛！
[root@linux tmp]# rm ./-aaa-
```

这是移除的指令（remove），相当于 dos 下的 del 指令！这里要注意的是，通常在 Linux 系统下，为了怕档案被误杀，所以很多 distributions 都已经预设有 -i 这个参数， -i 是指每个档案被杀掉之前都会让使用者确认一次，以预防误杀档案！而如果要连目录下的东西都一起杀掉的话， 例如子目录里面还有子目录时，那就要使用 -rf 这个参数了！不过，使用『 rm -rf 』这个指令之前，请千万注意了，因为，该目录或档案『肯定』会被 root 杀掉！因为系统不会再次询问你是否要砍掉呦！所以那是个超级严重的指令下达呦！ 得特别注意！不过，如果你确定该目录不要了，那么使用 rm -rf 来循环杀掉是不错的方式！

另外，范例三也是很有趣的例子，我们在之前就谈过，档名最好不要使用 "-" 号开头， 因为 "-" 后面接的是参数，因此，单纯的使用『 rm -aaa- 』系统的指令就会误判啦！ 那如果使用后面会谈到的正规表示法时，还是会出问题的！所以，只能用避过首位字符是 "-" 的方法啦！ 就是加上本目录『 ./ 』即可！如果 man rm 的话，其实还有一种方法，那就是『 rm -- -aaa- 』也可以啊！

---

* mv（移动档案与目录，或更名）

```
[root@linux ~]# mv [-fiu] source destination
[root@linux ~]# mv [options] source1 source2 source3 .... directory
参数：
-f  ：force 强制的意思，强制直接移动而不询问；
-i  ：若目标档案 (destination) 已经存在时，就会询问是否覆盖！
-u  ：若目标档案已经存在，且 source 比较新，才会更新 (update)
范例：
范例一：复制一档案，建立一目录，将档案移动到目录中
[root@linux ~]# cd /tmp
[root@linux tmp]# cp ~/.bashrc bashrc
[root@linux tmp]# mkdir mvtest
[root@linux tmp]# mv bashrc mvtest
# 将某个档案移动到某个目录去，就是这样做！

范例二：将刚刚的目录名称更名为 mvtest2
[root@linux tmp]# mv mvtest mvtest2 <== 这样就更名了！简单～
# 其实在 Linux 底下还有个有趣的指令，名称为 rename ，
# 该指令则专职进行档案的更名呢！用途也是不少～可以参阅 man rename 喔！

范例三：再建立两个档案，再全部移动到 /tmp/mvtest2 当中
[root@linux tmp]# cp ~/.bashrc bashrc1
[root@linux tmp]# cp ~/.bashrc bashrc2
[root@linux tmp]# mv bashrc1 bashrc2 mvtest2
# 注意到这边，如果有多个来源档案或目录，则最后一个目标文件一定是『目录！』
# 意思是说，将所有的数据移动到该目录的意思！
```

这是搬移 (move) 的意思！当你要移动档案或目录的时后，呵呵！这个指令就很重要啦！ 同样的，你也可以使用 -u (update)来测试新旧档案，看看是否需要搬移啰！ 另外一个用途就是『变更档名！』，我们可以很轻易的使用 mv 来变更一个档案的档名呢！不过，在 Linux 才有的指令当中，有个 rename ， 可以用来更改大量档案的档名，您可以利用 man rename 来查阅一下，也是挺有趣的指令喔！

取得路径的文件名称与目录名称

我们前面介绍的完整文件名 (包含目录名称与文件名称) 当中提到，完整档名最长可以到达 4096 个字符。那么您怎么知道那个是档名？那个是目录名？嘿嘿！就是利用斜线 (/) 来分辨啊！ 其实，取得文件名或者是目录名称，一般的用途应该是在写程序的时候，用来判断之用的啦～ 所以，这部分的指令可以用在第三篇内的 shell scripts 里头喔！ 底下我们简单的以几个范例来谈一谈 basename 与 dirname 的用途！

```
[root@linux ~]# basename /etc/sysconfig/network
network  <== 很简单！就取得最后的档名～
[root@linux ~]# dirname /etc/sysconfig/network
/etc/sysconfig  <== 取得的变成目录名了！
```

很简单的应用吧！

档案内容查阅：

刚刚我们提到的都只是在于显示档案的属性与权限，或者是移动与复制一个档案或目录而已， 那么如果我们要查阅一个档案的内容时，该如何是好呢？！这里有相当多有趣的指令可以来分享一下： 最常使用的显示档案内容的指令可以说是 cat 与 more 及 less 了！此外，如果我们要查看一个很大型的档案 (好几百 MB 时)，但是我们只需要后端的几行字而已，那么该如何是好？呵呵！用 tail 呀，此外， tac 这个指令也可以达到！好了，说说各个指令的用途吧！

- cat 由第一行开始显示档案内容
- tac 从最后一行开始显示，可以看出 tac 是 cat 的倒着写！
- nl 显示的时候，顺道输出行号！
- more 一页一页的显示档案内容
- less 与 more 类似，但是比 more 更好的是，他可以往前翻页！
- head 只看头几行
- tail 只看尾巴几行
- od 以二进制的方式读取档案内容！

直接检视档案内容

直接查阅一个档案的内容可以使用 cat/tac/nl 这几个指令啊！

- cat (concatenate)

```
[root@linux ~]# cat [-AEnTv]
参数：
-A  ：相当于 -vET 的整合参数，可列出一些特殊字符～
-E  ：将结尾的断行字符 $ 显示出来；
-n  ：打印出行号；
-T  ：将 [tab] 按键以 ^I 显示出来；
-v  ：列出一些看不出来的特殊字符
范例：
范例一：检阅 /etc/issue 这个档案的内容
[root@linux ~]# cat /etc/issue
Fedora Core release 4 (Stentz)
Kernel \r on an \m

范例二：承上题，顺便打印出行号时！
[root@linux ~]# cat -n /etc/issue
     1  Fedora Core release 4 (Stentz)
     2  Kernel \r on an \m
     3
# 看到了吧！可以印出行号呢！这对于大档案要找某个特定的行时，有点用处！
```

```
范例三：将 /etc/xinetd.conf 的内容完整的显示出来(包含特殊字符)
[root@linux ~]# cat -A /etc/xinetd.conf
#$
# Simple configuration file for xinetd$
#$
# Some defaults, and include /etc/xinetd.d/$
$
defaults$
{$
^Iinstances               = 60$
        log_type                = SYSLOG authpriv$
        log_on_success^I^I= HOST PID$
        log_on_failure^I^I= HOST$
^Icps^I^I^I= 25 30$
}$
$
includedir /etc/xinetd.d$
# 在一般的环境中，打印出来的结果在有 [tab] 与空格键，其实看不出来，
# 那么使用 cat -A 时，会将 [tab] 按键以 ^I 显示，而断行字符也会显示出来～
# 最特殊的当然就是断行字符了！这个段行字符在 Linux 与 Windows 是不一样的。
# 在 Linux 是以 $ 为断行字符，而在 Windows 则是以 ^M$ 为断行字符。
# 这部分我们会在 vi 软件的介绍时，再次的说明到喔！
```

嘿嘿！ Linux 里面有『猫』？！喔！不是的， cat 是 Concatenate （连续）的简写， 主要的功能是将一个档案的内容连续的印出在屏幕上面！例如上面的例子中，我们将 /etc/issue 印出来！如果加上 -n 的话，则每一行前面还会加上行号呦！鸟哥个人是比较少用 cat 啦！毕竟当你的档案内容的行数超过 40 行以上，嘿嘿！根本来不及看！所以，配合等一下要介绍的 more 或者是 less 来执行比较好！此外，如果是一般的 DOS 档案时，就需要特别留意一些奇奇怪怪的符号了， 例如断行与 [tab] 等，要显示出来，就得加入 -A 之类的参数了！

- tac (反向列示)

```
[root@linux ~]# tac /etc/issue

Kernel \r on an \m
Fedora Core release 4 (Stentz)
# 嘿嘿！与刚刚上面的范例一比较，是由最后一行先显示喔！
```

tac 这个好玩了！怎么说呢？详细的看一下， cat 与 tac ，有没有发现呀！对啦！ tac 刚好是将 cat 反写过来，所以他的功能就跟 cat 相反啦， cat 是由『第一行到最后一行连续显示在屏幕上』，而 tac 则是『 由最后一行到第一行反向在屏幕上显示出来 』，很好玩吧！

- nl (添加行号打印)

```
[root@linux ~]# nl [-bnw] 档案
参数：
-b  ：指定行号指定的方式，主要有两种：
      -b a ：表示不论是否为空行，也同样列出行号；
      -b t ：如果有空行，空的那一行不要列出行号；
-n  ：列出行号表示的方法，主要有三种：
      -n ln ：行号在屏幕的最左方显示；
      -n rn ：行号在自己字段的最右方显示，且不加 0 ；
      -n rz ：行号在自己字段的最右方显示，且加 0 ；
-w  ：行号字段的占用的位数。
范例：
范例一：列出 /etc/issue 的内容
[root@linux ~]# nl /etc/issue
     1  Fedora Core release 4 (Stentz)
     2  Kernel \r on an \m

# 注意看，这个档案其实有三行，第三行为空白(没有任何字符)，
# 因为他是空白行，所以 nl 不会加上行号喔！如果确定要加上行号，可以这样做：

[root@linux ~]# nl -b a /etc/issue
     1  Fedora Core release 4 (Stentz)
     2  Kernel \r on an \m
     3
# 呵呵！行号加上来啰～那么如果要让行号前面自动补上 0 呢？可这样

[root@linux ~]# nl -b a -n rz /etc/issue
000001  Fedora Core release 4 (Stentz)
000002  Kernel \r on an \m
000003
# 嘿嘿！自动在自己字段的地方补上 0 了～预设字段是六位数，如果想要改成 3 位数？

[root@linux ~]# nl -b a -n rz -w 3 /etc/issue
001     Fedora Core release 4 (Stentz)
002     Kernel \r on an \m
003
# 变成仅有 3 位数啰～
```

nl 可以将输出的档案内容自动的加上行号！其结果与 cat -n 有点不太一样， nl 可以将行号做比较多的显示设计，包括位数与是否自动补齐 0 等等的功能呢～

可翻页检视

前面提到的 nl 与 cat, tac 等等，都是一次性的将数据显示到屏幕上面，那有没有可以进行一页一页翻动的指令啊？ 让我们可以一页一页的观察，才不会前面的看不到啊～呵呵！有的！那就是 more 与 less 啰～

- more (一页一页翻动)

```
[root@linux ~]# more /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.5p
#
.......中间省略.......
                <== 重点在这一行喔!
```

仔细的给他看到上面的范例，如果 more 后面接的档案长度大于屏幕输出的行数时， 就会出现类似上面的图示。重点在最后一行，最后一行会显示出目前显示的百分比， 而且还可以在最后一行输入一些有用的指令喔！在 more 这个程序的运作过程中，你有几个按键可以按的：

- 空格键 (space)：代表向下翻一页；
- Enter　　　　：代表向下翻『一行』；
- /字符串　　　　：代表在这个显示的内容当中，向下搜寻『字符串』；
- :f　　　　　　：立刻显示出文件名以及目前显示的行数；
- q　　　　　　：代表立刻离开 more ，不再显示该档案内容。

要离开 more 这个指令的显示工作，可以按下 q 就能够离开了。而要向下翻页，就使用空格键即可。 比较有用的是搜寻字符串的功能，举例来说，我们使用『 more /etc/man.config 』来观察该档案， 若想要在该档案内搜寻 MANPATH 这个字符串时，可以这样做：

```
[root@linux ~]# more /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.5p
#
.......中间省略.......
/MANPATH   <== 输入了 / 之后，光标就会自动跑到最底下一行等待输入!
```

如同上面的说明，输入了 / 之后，光标就会跑到最底下一行，并且等待您的输入， 您输入了字符串之后，嘿嘿！ more 就会开始向下搜寻该字符串啰～而重复搜寻同一个字符串， 可以直接按下 n 即可啊！最后，不想要看了，就按下 q 即可离开 more 啦！

- less (一页一页翻动)

```
[root@linux ~]# less /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.5p
......中间省略........
:  <== 这里可以等待您输入指令!
```

less 的用法比起 more 又更加的有弹性，怎么说呢？在 more 的时候，我们并没有办法向前面翻， 只能往后面看，但若使用了 less 时，呵呵！就可以使用 [pageup] [pagedown] 等按键的功能来往前往后翻看文件，您瞧，是不是更容易使用来观看一个档案的内容了呢！？

除此之外，在 less 里头可以拥有更多的『搜寻』功能喔！不止可以向下搜寻，也可以向上搜寻～ 实在是很不错用～基本上，可以输入的指令有：

- 空格键　　：向下翻动一页；
- [pagedown]：向下翻动一页；
- [pageup]　：向上翻动一页；
- /字符串　　：向下搜寻『字符串』的功能；
- ?字符串　　：向上搜寻『字符串』的功能；
- n　　　　：重复前一个搜寻 (与 / 或 ? 有关！)
- N　　　　：反向的重复前一个搜寻 (与 / 或 ? 有关！)
- q　　　　：离开 less 这个程序；

查阅档案内容还可以进行搜寻的动作～瞧～ less 是否很不错用啊！ 其实 less 还有很多的功能喔！详细的使用方式请使用 man less 查询一下啊！ ^_^

资料撷取

我们可以将输出的资料作一个最简单的撷取，那就是取出前面 (head) 与取出后面 (tail) 文字的功能。不过，要注意的是， head 与 tail 都是以『行』为单位来进行数据撷取的喔！

- head (取出前面几行)

```
[root@linux ~]# head [-n number] 档案
参数：
-n  ：后面接数字，代表显示几行的意思
范例：
[root@linux ~]# head /etc/man.config
# 预设的情况中，显示前面十行！若要显示前 20 行，就得要这样：

[root@linux ~]# head -n 20 /etc/man.config
```

head 的英文意思就是『头』啦，那么这个东西的用法自然就是显示出一个档案的前几行啰！ 没错！就是这样！若没有加上 -n 这个参数时，预设只显示十行，若只要一行呢？那就加入『 head -n 1 filename 』即可！

- tail (取出后面几行)

```
[root@linux ~]# tail [-n number] 档案
参数：
-n  ：后面接数字，代表显示几行的意思
范例：
[root@linux ~]# tail /etc/man.config
# 预设的情况中，显示最后的十行！若要显示最后的 20 行，就得要这样：

[root@linux ~]# tail -n 20 /etc/man.config
```

那么有 head 自然就有 tail (尾巴) 啰！没错！这个 tail 的用法跟 head 的用法差不多类似，只是显示的是后面几行就是了！预设也是显示十行，若要显示非十行，就加 -n number 的参数！

> 例题一：假如我想要显示 ~/.bashrc 的第 11 到第 20 行呢？
> 答：
>
> 这个应该不算难，想一想，在第 11 到第 20 行，那么我取前 20 行，再取后十行，所以结果就是：『 head -n 20 ~/.bashrc | tail -n 10 』，这样就可以得到第 11 到第 20 行之间的内容了！但是里面涉及到管线命令，需要在第三篇的时候才讲的到！

非纯文字文件： od

我们上面提到的，都是在查阅纯文字文件 (ASCII 格式的档案) 的内容。 那么万一我们想要查阅非文字文件，举例来说，例如 /usr/bin/passwd 这个执行档的内容时， 又该如何去读出信息呢？事实上，由于执行档通常是 binary file ，使用上头提到的指令来读取他的内容时， 确实会产生类似乱码的数据啊！那怎么办？没关系，我们可以利用 od 这个指令来读取喔！

```
[root@linux ~]# od [-t TYPE] 档案
参数：
-t  ：后面可以接各种『类型 (TYPE)』的输出，例如：
      a       ：利用预设的字符来输出；
      c       ：使用 ASCII 字符来输出
      d[size] ：利用十进制(decimal)来输出数据，每个整数占用 size bytes ；
      f[size] ：利用浮点数值(floating)来输出数据，每个数占用 size bytes ；
      o[size] ：利用八进位(octal)来输出数据，每个整数占用 size bytes ；
      x[size] ：利用十六进制(hexadecimal)来输出数据，每个整数占用 size bytes ；
范例：
[root@linux ~]# od -t c /usr/bin/passwd
0000000 177   E   L   F 001 001 001  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000020 002  \0 003  \0 001  \0  \0  \0 260 225 004  \b   4  \0  \0  \0
```

```
0000040 020   E  \0  \0  \0  \0  \0  \0   4  \0      \0  \a  \0   (  \0
0000060 035  \0 034  \0 006  \0  \0  \0   4  \0  \0  \0   4 200 004  \b
0000100   4 200 004  \b 340  \0  \0  \0 340  \0  \0  \0 005  \0  \0  \0
.....中间省略.......
```

利用这个指令，可以将 data file 或者是 binary file 的内容数据给他读出来喔！ 虽然读出的来数值预设是使用非文字文件，亦即是 16 进位的数值来显示的， 不过，我们还是可以透过 -t c 的参数来将数据内的字符以 ASCII 类型的字符来显示， 虽然对于一般使用者来说，这个指令的用处可能不大，但是对于工程师来说， 这个指令可以将 binary file 的内容作一个大致的输出，他们可以看得出东西的啦～ ^_^

修改档案时间与建置新档： touch

我们在 ls 这个指令的介绍时，有稍微提到每个档案在 linux 底下都会记录三个主要的变动时间， 咦！那么三个时间是哪三个呢？

- modification time (mtime)：当该档案的『内容数据』变更时，就会更新这个时间！ 内容数据指的是档案的内容，而不是档案的属性喔！
- status time (ctime)：当该档案的『状态 (status)』改变时，就会更新这个时间，举例来说， 像是权限与属性被更改了，都会更新这个时间啊～
- access time (atime)：当『该档案的内容被取用』时，就会更新这个读取时间 (access)。 举例来说，我们使用 cat 去读取 ~/.bashrc ，就会更新 atime 了。

这是个挺有趣的现象，举例来说，我们来看一看您自己的 /etc/man.config 这个档案的时间吧！

```
[root@linux ~]# ls -l /etc/man.config
-rw-r--r--  1 root root 4506 Apr  8 19:11 /etc/man.config
[root@linux ~]# ls -l --time=atime /etc/man.config
-rw-r--r--  1 root root 4506 Jul 19 17:53 /etc/man.config
[root@linux ~]# ls -l --time=ctime /etc/man.config
-rw-r--r--  1 root root 4506 Jun 25 08:28 /etc/man.config
```

看到了吗？在预设的情况下， ls 显示出来的是该档案的 mtime ，也就是这个档案的内容上次被更动的时间。 至于我的系统是在 6/25 的时候安装的，因此，这个档案被产生但是状态被更动的时间就回溯到那个时间点了！ 而还记得刚刚我们使用的范例当中，有使用到这个档案啊，所以啊，他的 atime 就会变成刚刚使用的时间了！

档案的时间是很重要的，因为，如果档案的时间误判的话，可能会造成某些程序无法顺利的运作～ OK！那么万一我发现了一个档案来自未来(嘿嘿！不要怀疑！很多时候会有这个问题的！这个我们在安装的时候，提到的 GMT 时间就是那个意思啦～)，那该如何让该档案的时间变成『现在』的时刻呢？ 很简单啊！就用『touch』这个指令即可！

```
[root@linux ~]# touch [-acdmt] 档案
参数：
-a  ：仅修订 access time；
-c  ：仅修改时间，而不建立档案；
```

```
-d  ：后面可以接日期，也可以使用 --date="日期或时间"
-m  ：仅修改 mtime ；
-t  ：后面可以接时间，格式为[YYMMDDhhmm]
范例：

范例一：新建一个空的档案
[root@linux ~]# cd /tmp
[root@linux tmp]# touch testtouch
[root@linux tmp]# ls -l testtouch
-rw-r--r--  1 root root    0 Jul 19 20:49 testtouch
# 注意到，这个档案的大小是 0 呢！在预设的状态下，如果 touch 后面有接档案，
# 则该档案的三个时间 (atime/ctime/mtime) 都会更新为目前的时间。若该档案不存在，
# 则会主动的建立一个新的空的档案喔！例如上面这个例子！

范例二：将 ~/.bashrc 复制成为 bashrc，假设复制完全的属性，检查其日期
[root@linux tmp]# cp ~/.bashrc bashrc
[root@linux tmp]# ll bashrc; ll --time=atime bashrc; ll --time=ctime bashrc
-rwxr-xr-x  1 root root 395 Jul  4 11:45 bashrc  <==这是 mtime
-rwxr-xr-x  1 root root 395 Jul 19 20:44 bashrc  <==这是 atime
-rwxr-xr-x  1 root root 395 Jul 19 20:53 bashrc  <==这是 ctime
# 在这个案例当中，我们使用了 ; 这个指令分隔符，他的用法我们会在 Bash shell 中提到。
# 此外， ll 是 ls -l 的命令别名，这个我们也会在 bash shell 当中再次提及，
# 您目前可以简单的想成， ll 就是 ls -l 的简写即可！至于 ; 则是同时下达两个指令，
# 且让两个指令『依序』执行的意思。上面的结果当中我们可以看到，该档案变更的日期
# Jul 4 11:45，但是 atime 与 ctime 不一样啰～

范例三：修改案例二的 bashrc 档案，将日期调整为两天前
[root@linux tmp]# touch -d "2 days ago" bashrc
[root@linux tmp]# ll bashrc; ll --time=atime bashrc; ll --time=ctime bashrc
-rwxr-xr-x  1 root root 395 Jul 17 21:02 bashrc
-rwxr-xr-x  1 root root 395 Jul 17 21:02 bashrc
-rwxr-xr-x  1 root root 395 Jul 19 21:02 bashrc
# 跟上个范例比较看看，本来是 19 日的变成了 17 日了 (atime/mtime)～
# 不过， ctime 并没有跟着改变喔！

范例四：将上个范例的 bashrc 日期改为 2005/07/15 2:02
[root@linux tmp]# touch -t 0507150202 bashrc
[root@linux tmp]# ll bashrc; ll --time=atime bashrc; ll --time=ctime bashrc
-rwxr-xr-x  1 root root 395 Jul 15 02:02 bashrc
-rwxr-xr-x  1 root root 395 Jul 15 02:02 bashrc
-rwxr-xr-x  1 root root 395 Jul 19 21:05 bashrc
# 注意看看，日期在 atime 与 mtime 都改变了，但是 ctime 则是记录目前的时间！
```

透过 touch 这个指令，我们可以轻易的修订档案的日期与时间。并且，也可以建立一个空的档案喔！ 不过，要注意的是，即使我们复制一个档案时，复制所有的属性，但也没有办法复制 ctime 这个属性的。ctime 可以记录这个档案最近的状态 (status) 被改变的时间。无论如何，还是要告知大家， 我们平时看的档案属性中，比较重要的还是属于那个 mtime 啊！我们关心的常常是这个档案的『内容』 是什么时候被更动的说～瞭乎？

无论如何， touch 这个指令最常被使用的情况是：

- 建立一个空的档案；
- 将某个档案日期修订为目前 (mtime 与 atime)

档案与目录的预设权限与隐藏权限

由前一章的 Linux 档案属性 的内容我们可以知道一个档案有若干个属性，包括 (r, w, x) 等基本属性，及是否为目录 (d) 与档案 (-) 或者是连结档 (l) 等等的属性！那么要修改属性的方法在前面也约略提过了，这里再加强补充一下！ 此外，由于 Linux 还可以设定其它的系统安全属性，使用 chattr 来设定，而以 lsattr 来查看，最重要的属性就是可以设定其不可修改的特性！让连档案的拥有者都不能进行修改！这个属性可是相当重要的，尤其是在安全机制上面 (security)！

首先，先来复习一下上一章谈到的权限概念，将底下的例题看一看先～

例题二：你的系统有个一般身份使用者 dmtsai，他的群组为 users，他的家目录在 /home/dmtsai，你想将你的 ~/.bashrc 复制给他(假设你是 root)，可以怎么作？
答：

```
cp ~/.bashrc ~dmtsai/bashrc
chown dmtsai:users ~dmtsai/bashrc
```

在上面这个范例当中，我为了怕覆盖掉 dmtsai 自己的 ~dmtsai/.bashrc ，所以将档名更名了～ 而复制给他后，还要修正这个档案的拥有者与群组才行喔！

例题三：我想在 /tmp 底下建立一个目录，这个目录名称为 chap2_2_ex1 ，并且，这个目录拥有者为 dmtsai，群组为 users ，此外，任何人都可以进入该目录浏览档案，不过除了 dmtsai 之外，其它人都不能修改该目录下的档案。
答：

因为除了 dmtsai 之外，其它人不能修改该目录下的档案，此外， dmtsai 可以修改，所以整个目录的权限应该是 drwxr-xr-x 才对！因此

```
mkdir /tmp/chap2_2_ex1
chown -R dmtsai:users /tmp/chap2_2_ex1
chmod -R 755 /tmp/chap2_2_ex1
```

在上面这个例题当中，如果您知道 755 那个分数是怎么计算出来的，那么您应该对于权限有一定程度的概念了。 如果您不知道 755 怎么来的？那么.....赶快回去前一章看看 chmod 那个指令的介绍部分啊！这

部分很重要喔！您得要先清楚的了解到才行～否则就进行不下去啰～ 假设您对于权限都认识的差不多了，那么底下我们就要来谈一谈， 『新增一个档案或目录时，预设的权限是什么？』这个议题！

档案预设权限：umask

OK！那么现在我们知道如何建立或者是改变一个目录或档案的属性了，不过， 您知道当你建立一个新的档案或目录时，他的预设属性会是什么吗？呵呵！那就与 umask 这个玩意儿有关了！那么 umask 是在搞什么呢？基本上， umask 就是指定 『目前使用者在建立档案或目录时候的属性默认值』， 那么如何得知或设定 umask 呢？他的指定条件以底下的方式来指定：

```
[root@linux ~]# umask
0022
[root@linux ~]# umask -S
u=rwx,g=rx,o=rx
```

查阅的方式有两种，一种可以直接输入 umask ，就可以看到数字型态的权限设定分数， 一种则是加入 -S (Symbolic) 这个参数,就会以符号类型的方式来显示出权限了！奇怪的是,怎么 umask 会有四组数字啊？不是只有三组吗？是没错啦～ 第一组是特殊权限用的，我们先不要理他，所以先看后面三组即可。

在预设权限的属性上，目录与档案是不一样的。由于档案我们不希望他具有可执行的权力， 预设情况中，档案是没有可执行 (x) 权限的。因此：

- 若使用者建立为『档案』则预设『没有可执行 (x) 项目』，亦即只有 rw 这两个项目，也就是最大为 666 分，预设属性如下：
  -rw-rw-rw-
- 若使用者建立为『目录』，则由于 x 与是否可以进入此目录有关，因此预设为所有权限均开放，亦即为 777 分，预设属性如下：
  drwxrwxrwx

那么 umask 指定的是『该默认值需要减掉的权限！』因为 r、w、x 分别是 4、2、1 分，所以啰！也就是说，当要拿掉能写的权限，就是输入 2 分，而如果要拿掉能读的权限，也就是 4 分，那么要拿掉读与写的权限，也就是 6 分，而要拿掉执行与写入的权限，也就是 3 分，这样了解吗？请问您， 5 分是什么？呵呵！ 就是读与执行的权限啦！如果以上面的例子来说明的话，因为 umask 为 022 ，所以 user 并没有被拿掉属性，不过 group 与 others 的属性被拿掉了 2 (也就是 w 这个属性)，那么由于当使用者：

- 建立档案时：(-rw-rw-rw-) – (-----w--w-) ==> -rw-r--r--
- 建立目录时：(drwxrwxrwx) – (d----w--w-) ==> drwxr-xr-x

不相信吗？我们就来测试看看吧！

```
[root@linux ~]# umask
0022
[root@linux ~]# touch test1
[root@linux ~]# mkdir test2
```

```
[root@linux ~]# ll
-rw-r--r--  1 root root    0 Jul 20 00:36 test1
drwxr-xr-x  2 root root 4096 Jul 20 00:36 test2
```

呵呵！瞧见了吧？！确定属性是没有错的。好了，假如我们想要让与使用者同群组的人也可以存取档案呢？也就是说，假如 dmtsai 是 users 这个群组的人，而 dmtsai 作的档案希望让 users 同群组的人也可以存取，这也是常常被用在团队开发计划时，常常会考虑到的权限问题。在这样的情况下，我们的 umask 自然不能取消 group 的 w 权限，也就是说，我们希望制作出来的档案应该是 -rw-rw-r-- 的模样，所以啰，umask 应该是要 002 才好 (仅拿掉 others 的 w 权限)。那么如何设定 umask 呢？ 简单的很，直接在 umask 后面输入 002 就好了！

```
[root@linux ~]# umask 002
[root@linux ~]# touch test3
[root@linux ~]# mkdir test4
[root@linux ~]# ll
-rw-rw-r--  1 root root    0 Jul 20 00:41 test3
drwxrwxr-x  2 root root 4096 Jul 20 00:41 test4
```

所以说，这个 umask 对于档案与目录的预设权限是很有关系的！这个概念可以用在任何服务器上面， 尤其是未来在您架设档案服务器 (file server) ，举例来说， SAMBA Server 或者是 FTP server 时， 都是很重要的观念！这牵涉到您的使用者是否能够将档案进一步利用的问题喔！不要等闲视之！

例题四：假设您的 umask 为 003 ，请问该 umask 情况下，建立的档案与目录权限为？
答：

umask 为 003 ，所以拿掉的属性为 --------wx，因此：
档案： (-rw-rw-rw-) - (--------wx) = -rw-rw-r--
目录： (drwxrwxrwx) - (--------wx) = drwxrwxr--

Tips:
关于 umask 与权限的计算方式中，教科书喜欢使用二进制的方式来进行 AND 与 NOT 的计算， 不过，鸟哥还是比较喜欢使用符号方式来计算～联想上面比较容易一点～ 但是，有的书籍或者是 BBS 上面的朋友，有的人喜欢使用档案预设属性 666 与目录预设属性 777 来与 umask 进行相减的计算～这是不好的喔!以上面例题四的案例来看， 如果使用预设属性相加减，则档案变成：666-003=663，亦即是 -rw-rw--wx ，这可是完全不对的喔！ 想想看，原本档案就已经去除 x 的预设属性了，怎么可能突然间冒出来了？ 所以，这个地方得要特别小心喔！



在预设的情况中， root 的 umask 会拿掉比较多的属性，root 的 umask 预设是 022 ， 这是基于安全的考虑啦～至于一般身份使用者，通常他们的 umask 为 002 ，亦即保留同群组的写入权力！ 其实，关于预设 umask 的设定可以参考 /etc/bashrc 这个档案的内容，不过，不建议修改该档案， 您可以参考 bash shell 提到的环境参数设定档 (~/.bashrc) 的说明～这部分我们在第三章的时候会提到！

档案隐藏属性：

什么？档案还有隐藏属性？光是那九个权限就快要疯掉了，竟然还有隐藏属性，真是要命～ 但是没办法，就是有档案的隐藏属性存在啊！不过，这些隐藏的属性确实对于系统有很大的帮助的～ 尤其是在系统安全(Security) 上面，重要的紧呢！底下我们就来谈一谈如何设定与检查这些隐藏的属性吧！

- chattr (设定档案隐藏属性)

```
[root@linux ~]# chattr [+-=][ASacdistu] 档案或目录名称
参数：
+   ：增加某一个特殊参数，其它原本存在参数则不动。
-   ：移除某一个特殊参数，其它原本存在参数则不动。
=   ：设定一定，且仅有后面接的参数

A  ：当设定了 A 这个属性时，这个档案(或目录)的存取时间 atime (access)
     将不可被修改，可避免例如手提式计算机容易有磁盘 I/O 错误的情况发生！
S  ：这个功能有点类似 sync 的功能！就是会将数据同步写入磁盘当中！
     可以有效的避免数据流失！
a  ：当设定 a 之后，这个档案将只能增加数据，而不能删除，只有 root
     才能设定这个属性。
c  ：这个属性设定之后，将会自动的将此档案『压缩』，在读取的时候将会自动解压缩，
     但是在储存的时候，将会先进行压缩后再储存(看来对于大档案似乎蛮有用的！)
d  ：当 dump(备份)程序被执行的时候，设定 d 属性将可使该档案(或目录)不具有 dump 功能
i  ：这个 i 可就很厉害了！他可以让一个档案『不能被删除、改名、设定连结也无法写入
     或新增资料！』对于系统安全性有相当大的帮助！
j  ：当使用 ext3 这个档案系统格式时，设定 j 属性将会使档案在写入时先记录在
     journal 中！但是当 filesystem 设定参数为 data=journalled 时，由于已经设定了
     日志了，所以这个属性无效！
s  ：当档案设定了 s 参数时，他将会被完全的移除出这个硬盘空间。
u  ：与 s 相反的，当使用 u 来设定档案时，则数据内容其实还存在磁盘中，
     可以使用来 undeletion.
注意：这个属性设定上面，比较常见的是 a 与 i 的设定值，而且很多设定值必须要身为
root 才能够设定的喔！
范例：
[root@linux ~]# cd /tmp
[root@linux tmp]# touch attrtest
[root@linux tmp]# chattr +i attrtest
[root@linux tmp]# rm attrtest
rm: remove write-protected regular empty file `attrtest'? y
rm: cannot remove `attrtest': Operation not permitted
# 看到了吗？呼呼！连 root 也没有办法将这个档案删除呢！赶紧解除设定！
[root@linux tmp]# chattr -i attrtest
```

这这个指令是重要的，尤其是在系统的安全性上面！由于这些属性是隐藏的性质，所以需要以 lsattr 才能看到该属性呦！其中，个人认为最重要的当属 +i 这个属性了，因为他可以让一个档案无法被更动，对

于需要强烈的系统安全的人来说，真是相当的重要的!里头还有相当多的属性是需要 root 才能设定的呢!此外，如果是 log file 这种的登录档，就更需要 +a 这个可以增加，但是不能修改旧有的数据与删除的参数了！ 怎样？很棒吧！未来提到登录档的认知时，我们再来聊一聊如何设定他吧!

- lsattr (显示档案隐藏属性)

```
[root@linux ~]# lsattr [-aR] 档案或目录
参数:
-a : 将隐藏文件的属性也秀出来;
-R : 连同子目录的数据也一并列出来!
范例:
[root@linux tmp]# chattr +aij attrtest
[root@linux tmp]# lsattr
----ia---j--- ./attrtest
```

使用 chattr 设定后，可以利用 lsattr 来查阅隐藏的属性。不过， 这两个指令在使用上必须要特别小心，否则会造成很大的困扰。例如：某天你心情好，突然将 /etc/shadow 这个重要的密码记录档案给他设定成为具有 i 的属性，那么过了若干天之后， 你突然要新增使用者，却一直无法新增!别怀疑，赶快去将 i 的属性拿掉吧!

档案特殊权限： SUID/SGID/Sticky Bit

我们前面一直提到关于档案的重要权限，那就是 rwx 这三个读、写、执行的权限。 但是，眼尖的朋友们一定注意到了一件事，那就是，怎么我们的 /tmp 权限怪怪的？ 还有，那个 /usr/bin/passwd 也怪怪的？怎么回事啊？看看先：

```
[root@linux ~]# ls -ld /tmp ; ls -l /usr/bin/passwd
drwxrwxrwt  5 root root 4096 Jul 20 10:00 /tmp
-r-s--x--x  1 root root 18840 Mar  7 18:06 /usr/bin/passwd
```

不是只有 rwx 吗？还有其它的特殊权限啊？啊.....头又开始昏了～ @_@ 呵呵，不要担心啦，我们这里先不谈这两个权限，只是先介绍一下而已。 因为要了解这几个特殊的权限，必须要先具有账号的 ID 概念，以及程序的程序 (process) 概念后， 才能够进一步的了解这个特殊权限所代表的意义。

- Set UID

会制作出 s 与 t 的权限，是为了让一般使用者在执行某些程序的时候， 能够暂时的具有该程序拥有者的权限。 举例来说好了，我们知道账号与密码的存放档案其实是 /etc/passwd 与 /etc/shadow 。 而 /etc/shadow 这个档案的权限是什么？是『-r--------』。且他的拥有者是 root 喔！在这个权限中，仅有 root 可以『强制』储存，其它人是连看都没有办法看的呐!

但是偏偏鸟哥使用 dmtsai 这个一般身份使用者去更新自己的密码时，使用的就是 /usr/bin/passwd 这个程序， 却是可以更新自己的密码的，也就是说， dmtsai 这个一般身份使用者可以存取 /etc/shadow 这个密码文件！但!怎么可能？明明 /etc/shadow 就是没有 dmtsai 可以存取的权限啊!呵呵～这就是 s 这个权限的帮助啦！ 当 s 这个权限在 user 的 x 时，也就是类似上表的 -r-s--x--x ，称为 Set UID ，简称为 SUID ， 这个 UID 代表的是 User 的 ID ，而 User 代表的则是这个程序 (/usr/bin/passwd) 的拥有者 (root 啊！)。 那么由上面的定义中，我们知道了，当 dmtsai 这个使用者执行 /usr/bin/passwd

时，他就会『暂时』的得到档案拥有人 root 的权限。

SUID 仅可用在『二进制制档案(binary file)』上， SUID 因为是程序在执行的过程中拥有档案拥有者的权限，因此，他仅可用于 binary file ， 不能够用在批次档 (shell script) 上面的！这是因为 shell script 只是将很多的 binary 执行档叫进来执行而已！所以 SUID 的权限部分，还是得要看 shell script 呼叫进来的程序的设定， 而不是 shell script 本身。当然，SUID 对于目录也是无效的～这点要特别留意。

- Set GID

进一步来说，如果 s 的权限是在 group 时，那么就是 Set GID ，简称为 SGID。 SGID 可以用在两个部分喔！

  - 档案：如果 SGID 是设定在 binary file 上面，则不论使用者是谁，在执行该程序的时候， 他的有效群组 (effective group) 将会变成该程序的群组所有人 (group id)。

  - 目录：如果 SGID 是设定在 A 目录上面，则在该 A 目录内所建立的档案或目录的 group ，将会是 此 A 目录的 group ！

一般来说， SGID 应该是比较多用在特定的多人团队的项目开发上， 在系统当中应该会比较少这个设定才对～

- Sticky Bit

这个 Sticky Bit 目前只针对目录有效，对于档案已经没有效果了。 SBit 对于目录的作用是：『在具有 SBit 的目录下，使用者若在该目录下具有 w 及 x 的权限， 则当使用者在该目录下建立档案或目录时，只有档案拥有者与 root 才有权力删除』。换句话说：当甲这个使用者于 A 目录下是拥有 group 或者是 other 的项目，并且拥有 w 的权限， 这表示『甲使用者对该目录内任何人建立的目录或档案均可进行 "删除/更名/搬移" 等动作。』 不过，如果将 A 目录加上了 Sticky bit 的权限项目时， 则甲只能够针对自己建立的档案或目录进行删除/更名/移动等动作。

举例来说，我们的 /tmp 本身的权限是『drwxrwxrwt』， 在这样的权限内容下，任何人都可以在 /tmp 内新增、修改档案，但仅有该档案/目录建立者与 root 能够删除自己的目录或档案。这个特性也是挺重要的啊！你可以这样做个简单的测试：

1. 以 root 登入系统，并且进入 /tmp 当中；
2. touch test，并且更改 test 权限成为 777 ；
3. 以一般使用者登入，并进入 /tmp；
4. 尝试删除 test 这个档案！

更多关于 SUID/SGID/Sticky Bit 的介绍，我们会在 程序与资源管理 中再次提及的，目前，您先有个简单的基础概念就好了！当然，也可以参考鸟园讨论区的这一篇讨论：
http://phorum.vbird.org/viewtopic.php?t=20256

- SUID/SGID/SBIT 权限设定

前面介绍过 SUID 与 SGID 的功能，那么如何开启档案使成为具有 SUID 与 SGID 的权限呢？！ 这就需要刚刚的数字更改权限的方法了！现在你应该已经知道数字型态个更改权限方式为『三个数字』的组合， 那么如果在这三个数字之前再加上一个数字的话，那最前的面数字就代表这几个属性了！ (注：通常我们使

用 chmod xyz filename 的方式来设定 filename 的属性时，则是假设没有 SUID, SGID 及 Sticky bit 啦！ ）

- 4 为 SUID
- 2 为 SGID
- 1 为 Sticky bit

假设要将一个档案属性改为『-rwsr-xr-x』时，由于 s 在使用者权限中，所以是 SUID ，因此， 在原先的 755 之前还要加上 4 ，也就是：『 chmod 4755 filename 』来设定！此外，还有大 S 与大 T 的产生喔！参考底下的范例啦！(注意：底下的范例只是练习而已， 所以鸟哥使用同一个档案来设定，您必须了解 SUID 不是用在目录上，而 SBIT 不是用在档案上的喔！)

```
[root@linux ~]# cd /tmp
[root@linux tmp]# touch test
[root@linux tmp]# chmod 4755 test; ls -l test
-rwsr-xr-x  1 root root 0 Jul 20 11:27 test
[root@linux tmp]# chmod 6755 test; ls -l test
-rwsr-sr-x  1 root root 0 Jul 20 11:27 test
[root@linux tmp]# chmod 1755 test; ls -l test
-rwxr-xr-t  1 root root 0 Jul 20 11:27 test
[root@linux tmp]# chmod 7666 test; ls -l test
-rwSrwSrwT  1 root root 0 Jul 20 11:27 test
# 这个例子就要特别小心啦！怎么会出现大写的 S 与 T 呢？不都是小写的吗？
# 因为 s 与 t 都是取代 x 这个参数的，但是你有没有发现阿，我们是下达
# 7666 喔！也就是说， user, group 以及 others 都没有 x 这个可执行的标志
# ( 因为 666 嘛！ )，所以，这个 S, T 代表的就是『空的』啦！怎么说？
# SUID 是表示『该档案在执行的时候，具有档案拥有者的权限』，但是档案
# 拥有者都无法执行了，哪里来的权限给其它人使用？当然就是空的啦！ ^_^
```

档案类型：file

如果你想要知道某个档案的基本数据，例如是属于 ASCII 或者是 data 档案，或者是 binary ， 且其中有没有使用到动态函式库 (share library) 等等的信息，就可以利用 file 这个指令来检阅喔！ 举例来说：

```
[root@linux ~]# file ~/.bashrc
/root/.bashrc: ASCII text  <== 告诉我们是 ASCII 的纯文字文件啊！
[root@linux ~]# file /usr/bin/passwd
/usr/bin/passwd: setuid ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), for GNU/Linux 2.2.5, dynamically linked (uses shared libs), stripped
# 数据可多了～包括这个日 Set UID 2 的档案，使用 shared libs,
# 适合于 Intel 的 386 以上机种的硬件，很清楚吧！
[root@linux ~]# file /var/lib/slocate/slocate.db
/var/lib/slocate/slocate.db: data  <== 这是 data 档案！
```

透过这个指令，我们可以简单的先判断这个档案的格式为何喔！

档案的搜寻：

档案的搜寻可就厉害了!因为我们常常需要知道那个档案放在哪里,所以来谈一谈怎么搜寻吧!在 Linux 底下也有相当优异的搜寻系统呦！通常 find 不很常用的！因为速度慢之外，也很操硬盘！通常我们都是先使用 whereis 或者是 locate 来检查,如果真的找不到了,才以 find 来搜寻呦!为什么呢？因为 whereis 与 locate 是利用数据库来搜寻数据，所以相当的快速，而且并没有实际的搜寻硬盘，比较省时间啦！

- which（寻找『执行档』）

```
[root@linux ~]# which [-a] command
参数：
-a ：将所有可以找到的指令均列出，而不止第一个被找到的指令名称
范例：
[root@linux ~]# which passwd
/usr/bin/passwd
[root@linux ~]# which traceroute -a
/usr/sbin/traceroute
/bin/traceroute
```

这个指令是根据『PATH』这个环境变量所规范的路径，去搜寻『执行档』的档名～ 所以，重点是找出『执行档』而已！且 which 后面接的是『完整档名』喔！若加上 -a 参数，则可以列出所有的可以找到的同名执行文件，而非仅显示第一个而已！

- whereis（寻找特定档案）

```
[root@linux ~]# whereis [-bmsu] 档案或目录名
参数：
-b    :只找 binary 的档案
-m    :只找在说明文件 manual 路径下的档案
-s    :只找 source 来源档案
-u    :没有说明档的档案！
范例：
[root@linux ~]# whereis passwd
passwd: /usr/bin/passwd /etc/passwd /etc/passwd.OLD
/usr/share/man/man1/passwd.1.gz /usr/share/man/man5/passwd.5.gz
# 任何与 passwd 有关的档名都会被列出来～

[root@linux ~]# whereis -b passwd
passwd: /usr/bin/passwd /etc/passwd /etc/passwd.OLD

[root@linux ~]# whereis -m passwd
passwd: /usr/share/man/man1/passwd.1.gz /usr/share/man/man5/passwd.5.gz
```

等一下我们会提到 find 这个搜寻指令， find 是很强大的搜寻指令，但时间花用的很大！（因为 find 是直接搜寻硬盘，为如果你的硬盘比较老旧的话，嘿嘿！有的等的！）这个时候 whereis 就相当的好用了！另外， whereis 可以加入参数来找寻相关的资料， 例如如果你是要找可执行档 (binary) 那么加上 -b 就可以啦！例如上面的范例针对 passwd 这支程序来说明！如果不加任何参数的话，那么就将所有的数据列出来啰！

那么 whereis 到底是使用什么咚咚呢？为何搜寻的速度会比 find 快这么多？！ 其实那也没有什么！这是因为 Linux 系统会将系统内的所有档案都记录在一个数据库档案里面， 而当使用 whereis 或者是底下要说的 locate 时，都会以此数据库档案的内容为准， 因此，有的时后你还会发现使用这两个执行档时，会找到已经被杀掉的档案！ 而且也找不到最新的刚刚建立的档案呢！ 这就是因为这两个指令是由数据库当中的结果去搜寻档案的所在啊！

另外，基本上 Linux 每天会针对 Linux 主机上所有档案的所在进行搜寻数据库的更新， 更新的程序就是 updatedb ，你可以在 FC4 系统的 /etc/cron.daily/slocate.cron 这个档案找到相关的机制呦！当然，也可以直接使用 /usr/bin/updatedb 来更新数据库档案呢！

- locate

```
[root@linux ~]# locate filename
[root@linux ~]# locate passwd
/lib/security/pam_passwdqc.so
/lib/security/pam_unix_passwd.so
/usr/lib/kde3/kded_kpasswdserver.so
/usr/lib/kde3/kded_kpasswdserver.la
.......中间省略.......
```

这个 locate 的使用更简单，直接在后面输入『档案的部分名称』后，就能够得到结果。 举上面的例子来说，我输入 locate passwd ，那么在完整文件名 (包含路径名称) 当中，只要有 passwd 在其中， 就会被显示出来的！这也是个很方便好用的指令，如果您忘记某个档案的完整档名时～～

但是,这个东西还是有使用上的限制呦!为什么呢？您会发现使用 locate 来寻找数据的时候特别的快，这是因为 locate 寻找的数据是由『已建立的数据库 /var/lib/slocate/』 里面的数据所搜寻到的，所以不用直接在去硬盘当中存取数据，呵呵！当然是很快速啰！ 那么有什么限制呢？就是因为他是经由数据库来搜寻的，而数据库的建立预设是在每天执行一次 (每个 distribution 都不同， FC4 是每天更新数据库一次！)，所以当您新建立起来的档案， 却还在数据库更新之前搜寻该档案，那么 locate 会告诉您『找不到！』呵呵！因为必须要更新数据库呀！

那么我到底要建立哪些数据库呢？是否全部都要建立？似乎不需要，这个时候， 你可以自己选择需要建立档案数据库的目录呢！你可以在 /etc/updatedb.conf 这个档案内设定。 建议您使用默认值就好了，不过，在 /etc/updatedb.conf 里面，请把『DAILY_UPDATE=no』改成 『DAILY_UPDATE=yes』就好了。至于修改的方法等到我们第三章提完 vi 后，您就会晓得啰～ 当然啦，也可以自行手动执行 updatedb 即可！

- find

```
[root@linux ~]# find [PATH] [option] [action]
```

```
参数：
1. 与时间有关的参数：
   -atime n ：n 为数字，意义为在 n 天之前的『一天之内』被 access 过的档案；
   -ctime n ：n 为数字，意义为在 n 天之前的『一天之内』被 change 过状态的档案；
   -mtime n ：n 为数字，意义为在 n 天之前的『一天之内』被 modification 过的档案；
   -newer file ：file 为一个存在的档案，意思是说，只要档案比 file 还要新，
                就会被列出来～
2. 与使用者或群组名称有关的参数：
   -uid n ：n 为数字，这个数字是使用者的账号 ID，亦即 UID ，这个 UID 是记录在
           /etc/passwd 里面与账号名称对应的数字。这方面我们会在第四篇介绍。
   -gid n ：n 为数字，这个数字是群组名称的 ID，亦即 GID，这个 GID 记录在
           /etc/group，相关的介绍我们会第四篇说明～
   -user name ：name 为使用者账号名称喔！例如 dmtsai
   -group name：name 为群组名称喔，例如 users ；
   -nouser    ：寻找档案的拥有者不存在 /etc/passwd 的人！
   -nogroup   ：寻找档案的拥有群组不存在于 /etc/group 的档案！
               当您自行安装软件时，很可能该软件的属性当中并没有档案拥有者，
               这是可能的！在这个时候，就可以使用 -nouser 与 -nogroup 搜寻。
3. 与档案权限及名称有关的参数：
   -name filename：搜寻文件名称为 filename 的档案；
   -size [+-]SIZE：搜寻比 SIZE 还要大(+)或小(-)的档案。这个 SIZE 的规格有：
                  c: 代表 byte， k: 代表 1024bytes。所以，要找比 50KB
                  还要大的档案，就是『 -size +50k 』
   -type TYPE    ：搜寻档案的类型为 TYPE 的，类型主要有：一般正规档案 (f),
                  装置档案 (b, c), 目录 (d), 连结档 (l), socket (s),
                  及 FIFO (p) 等属性。
   -perm mode  ：搜寻档案属性『刚好等于』 mode 的档案，这个 mode 为类似 chmod
                的属性值，举例来说， -rwsr-xr-x 的属性为 4755 ！
   -perm -mode ：搜寻档案属性『必须要全部囊括 mode 的属性』的档案，举例来说，
                我们要搜寻 -rwxr--r-- ，亦即 0744 的档案，使用 -perm -0744，
                当一个档案的属性为 -rwsr-xr-x ，亦即 4755 时，也会被列出来，
                因为 -rwsr-xr-x 的属性已经囊括了 -rwxr--r-- 的属性了。
   -perm +mode ：搜寻档案属性『包含任一 mode 的属性』的档案，举例来说，我们搜寻
                -rwxr-xr-x ，亦即 -perm +755 时，但一个档案属性为 -rw-------
                也会被列出来，因为他有 -rw.... 的属性存在！
4. 额外可进行的动作：
   -exec command ：command 为其它指令，-exec 后面可再接额外的指令来处理搜寻到
                  的结果。
   -print        ：将结果打印到屏幕上，这个动作是预设动作！
范例：
范例一：将过去系统上面 24 小时内有更动过内容 (mtime) 的档案列出
[root@linux ~]# find / -mtime 0
# 那个 0 是重点！0 代表目前的时间，所以，从现在开始到 24 小时前，
```

```
# 有变动过内容的档案都会被列出来！那如果是三天前的 24 小时内？
# find / -mtime 3 ，意思是说今天之前的 3*24 ~ 4*24 小时之间
# 有变动过的档案都被列出的意思！同时 -atime 与 -ctime 的用法相同。

范例二：寻找 /etc 底下的档案，如果档案日期比 /etc/passwd 新就列出
[root@linux ~]# find /etc -newer /etc/passwd
# -newer 用在分辨两个档案之间的新旧关系是很有用的！

范例三：搜寻 /home 底下属于 dmtsai 的档案
[root@linux ~]# find /home -user dmtsai
# 这个东西也很有用的～当我们要找出任何一个使用者在系统当中的所有档案时，
# 就可以利用这个指令将属于某个使用者的所有档案都找出来喔！

范例四：搜寻系统中不属于任何人的档案
[root@linux ~]# find / -nouser
# 透过这个指令，可以轻易的就找出那些不太正常的档案。
# 如果有找到不属于系统任何人的档案时，不要太紧张，
# 那有时候是正常的～尤其是您曾经以原始码自行编译软件时。

范例五：找出档名为 passwd 这个档案
[root@linux ~]# find / -name passwd
# 利用这个 -name 可以搜寻档名啊！

范例六：搜寻档案属性为 f (一般档案) 的档案
[root@linux ~]# find /home -type f
# 这个 -type 的属性也很有帮助喔！尤其是要找出那些怪异的档案，
# 例如 socket 与 FIFO 档案，可以用 find /var -type p 或 -type s 来找！

范例七：搜寻档案当中含有 SGID/SUID/SBIT 的属性
[root@linux ~]# find / -perm +7000
# 所谓的 7000 就是 ---s--s--t ，那么只要含有 s 或 t 的就列出，
# 所以当然要使用 +7000 ，使用 -7000 表示要含有 ---s--s--t 的所有三个权限，
# 因此，就是 +7000 ～瞭乎？

范例八：将上个范例找到的档案使用 ls -l 列出来～
[root@linux ~]# find / -perm +7000 -exec ls -l {} \;
# 注意到，那个 -exec 后面的 ls -l 就是额外的指令，
# 而那个 {} 代表的是『由 find 找到的内容』的意思～所以， -exec ls -l {}
# 就是将前面找到的那些档案以 ls -l 列出长的数据！至于 \; 则是表示
# -exec 的指令到此为止的意思～意思是说，整个指令其实只有在
# -exec (里面就是指令下达) \;
# 也就是说，-exec 最后一定要以 \; 结束才行！这样了解了吗？！
```

```
范例九：找出系统中，大于 1MB 的档案
[root@linux ~]# find / -size +1000k
# 虽然在 man page 提到可以使用 M 与 G 分别代表 MB 与 GB，
# 不过，俺却试不出来这个功能～所以，目前应该是仅支持到 c 与 k 吧！
```

如果你要寻找一个档案的话，那么使用 find 会是一个不错的主意！ 他可以根据不同的参数来给予档案的搜寻功能！例如你要寻找一个档名为 httpd.conf 的档案，你知道他应该是在 /etc 底下，那么就可以使用『 find /etc -name httpd.conf 』噜！ 那如果你记得有一个档案档名包含了 httpd ，但是不知道全名怎办？！呵呵，就用万用字符 * 吧，如上以：『 find /etc -name '*httpd*' 』就可将档名含有 httpd 的档案都列出来啰！不过，由于 find 在寻找数据的时后相当的耗硬盘！所以没事情不要使用 find 啦！有更棒的指令可以取代呦！那就是 whereis 与 locate 啰！！

但，不管怎么说， find 在找寻特殊的档案属性，以及特殊的档案权限 (SUID/SGID等等) 时， 是相当有用的工具程序之一！重要重要！

本章习题练习：
(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 什么是绝对路径与相对路径

  绝对路径的写法为由 / 开始写，至于相对路径则不由 / 开始写！此外，相对路径为相对于目前工作目录的路径！

- 如何更改一个目录的名称？例如由 /home/test 变为 /home/test2

  mv /home/test /home/test2

- PATH 这个环境变量的意义？

  这个是用来指定执行档执行的时候，档案搜寻的目录路径。

- umask 有什么用处与优点？

  umask 可以拿掉一些属性，因此，适当的定义 umask 有助于系统的安全， 因为他可以用来建立预设的目录或档案的权限。

- 当一个使用者的 umask 分别为 033 与 044 他所建立的档案与目录的权限为何？

  在 umask 为 033 时，则预设是拿掉 group 与 other 的 w(2)x(1) 权限，因此权限就成为『档案 -rw-r--r-- ， 目录 drwxr--r-- 』而当 umask 044 时，则拿掉 r 的属性，因此就成为『档案 -rw--w--w-，目录 drwx-wx-wx』

- 什么是 SUID ？

  当一个指令具有 SUID 的功能时，则当其它人使用这个指令时，该程序将具有指令拥有者的权限。

- 当我要查询 /usr/bin/passwd 这个档案的一些属性时，可以使用什么指令来查询?

  ls -al, file, lsattr

- 尝试用 find 找出目前 linux 系统中，所有具有 SUID 的档案有哪些?

  find / -type f -perm -4000 -print

档案的压缩与打包

最近更新日期：2005/08/29

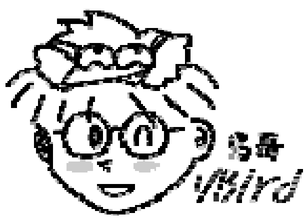
在 Linux 底下有相当多的压缩指令可以运作喔！这些压缩指令可以让我们更方便从网络上面下载大型的档案呢！ 此外，我们知道在 Linux 底下的附档名是没有什么很特殊的意义的，不过，针对这些压缩指令所做出来的压缩档， 为了方便记忆，还是会有一些特殊的命名方式啦！就让我们来看看吧！

压缩档案的用途与技术：

您是否有过文件档案太大，导致无法以一片软盘将他复制完成的困扰？ 又，您是否有过，发现一个软件里面有好多档案，这些档案要将他复制与携带都很不方便的问题？ 还有，您是否有过要备份某些重要数据，偏偏这些数据量太大了，耗掉了你很多的硬盘与磁盘空间呢？ 这个时候，那个好用的『档案压缩』技术可就派的上用场了！ 因为这些比较大型的档案透过所谓的档案压缩技术之后，可以将他的磁盘使用量降低，可以达到减低档案容量的效果，此外，有的压缩程序还可以进行容量限制， 使一个大型档案可以分割成为数个小型档案，以方便软盘片携带呢！

那么什么是『档案压缩』呢？我们来稍微谈一谈他的原理好了。目前我们使用的计算机系统中都是使用所谓的 bytes 单位来计量的！不过，事实上，计算机最小的计量单位应该是 bits 才对啊，此外，我们也知道 1 byte = 8 bits 。但是如果今天我们只是记忆一个数字，亦即是 1 这个数字呢？他会如何记录？假设一个 byte 可以看成底下的模样：

□□□□□□□□



> Tips:
> 由于 1 byte = 8 bits ，所以每个 byte 当中会有 8 个空格，而每个空格可以是 0, 1 ，这里仅是做为一个约略的介绍，读者不必刻意记忆。

而由于我们记录数字是 1 ，考虑计算机所谓的二进制喔，如此一来， 1 会在最右边占据 1 个 bit ，而其它的 7 个 bits 将会自动的被填上 0 啰！你看看，其实在这样的例子中，那 7 个 bits 应该是『空的』才对！不过，为了要满足目前我们的操作系统数据的存取，所以就会将该数据转为 byte 的型态来记录了！而一些聪明的计算机工程师就利用一些复杂的计算方式， 将这些没有使用到的空间『丢』出来，以让档案占用的空间变小！这就是压缩的技术啦！

简单的说，你可以将他想成，其实档案里面有相当多的『空间』存在，并不是完全填满的， 而『压缩』的技术就是将这些『空间』填满，以让整个档案占用的容量下降！ 不过，这些『压缩过的档案』并无法直接被我们的操作系统所使用的，因此， 若要使用这些被压缩过的档案数据，则必须将他『还原』回来未压缩前的模样， 那就是所谓的『解压缩』啰！而至于压缩前与压缩后的档案所占用的磁盘空间大小， 就可以被称为是『压缩比』啰！更多的技术文件或许你可以参考一下：

- RFC 1952 文件：http://www.faqs.org/rfcs/rfc1952.html
- 鸟哥站上的备份：http://linux.vbird.org/linux_basic/0240tarcompress/0240tarcompress_gzip.php

这个『压缩』与『解压缩』的动作有什么好处呢？最大的好处就是压缩过的档案容量变小了， 所以你的硬盘容量无形之中就可以容纳更多的数据，此外，在一些网络数据的传输中，也会由于数据量的降低， 好让网络频宽可以用来作更多的工作！而不是老是卡在一些大型的档案上面呢！目前很多的 WWW 网站也是利用档案压缩的技术来进行数据的传送，好让网站的可利用率上升喔！

> Tips:
> 这种技术蛮有趣的！他让您网站上面『看的到的数据』在经过网络传输时，使用的是『压缩过的数据』， 等到这些压缩过的数据到达你的计算机主机时，再进行解压缩，由于目前的计算机运算速度相当的快速， 因此其实在网页浏览的时候，时间都是花在『数据的传输』上面，而不是 CPU 的运算啦！，如此一来，由于压缩过的数据量降低了，自然传送的速度就会增快不少！



若您是一位软件工程师，那么相信您也会喜欢将你自己的软件压缩之后提供大家下载来使用， 毕竟没有人喜欢自己的网站天天都是频宽满载的吧？！举个例子来说， Linux 2.4.19 完整的核心大小约有 200 MB 左右，而由于核心主要多是 ASCII code 的纯文字型态档案，这种档案的『多余空间』最多了。而一个提供下载的压缩过的 2.4.19 核心大约仅有 30MB 左右，差了几倍呢？您可以自己算一算喔！

Linux 系统常见的压缩指令：

如果您常常在网络上面捉 Linux 的数据下来玩的话，大概会晓得的是，这些供人下载的档案通常都是『压缩』过的！ 为了什么？上面已经稍微提过啦！呵呵！压缩过的档案具有节省频宽、节省磁盘空间等等的优点，并且还方便携带呢！ ^_^ ！而，您应该也会知道，这些被压缩过的档案，通常其附档名都是『 *.tar, *.tar.gz, *.tgz, *.gz, *.Z, *.bz2 』 等等的，为什么要订定这些压缩档案附档名为这样的模样呢？

这是因为在 Linux 上面压缩的指令相当的多，并且，这些压缩指令可能无法针对每种压缩档案都可以解的开， 毕竟目前的压缩技术五花八门，每种压缩计算的方法都不是完全相同的，所以啰，当你捉到某个压缩档时， 自然就需要知道压缩他的是那个指令啦，好用来对照着解压缩啊！ ^_^！也就是说，虽然 Linux 档案的属性基本上是与文件名没有绝对关系的，能不能执行与他的档案属性有关而已， 与档名的关系很小！但是，为了帮助我们小小的人类脑袋瓜子，所以适当的文件名称附档名还是必要的！ 因此，目前就有一些常常见到的压缩档案的附档名啦！我们仅列出常见的几样在底下，给大家权做参考之用：

- *.Z　　compress 程序压缩的档案；
- *.bz2　bzip2 程序压缩的档案；
- *.gz　 gzip 程序压缩的档案；

- *.tar tar 程序打包的数据，并没有压缩过；
- *.tar.gz tar 程序打包的档案，其中并且经过 gzip 的压缩

目前常见的压缩程序主要就是如同上面提到的附档名对应的那些指令啦！最早期的要算是 compress 这个家伙了，不过这个 compress 指令目前已经不再是预设的压缩软件了～ 而后，后来的 GNU 计划开发出新一代的压缩指令 gzip (GNU zip) 用来取代 compress 这个老牌的压缩指令，再来还有 bzip2 这个压缩比更好的压缩指令呢！不过，这些指令通常仅能针对一个档案来压缩与解压缩，如此一来， 每次压缩与解压缩都要一大堆档案，岂不烦人？此时，那个所谓的『打包软件』就显的很重要啦！

在 Unix-Like 当中，有个软件很好玩，他就是 tar 这支程序！这个 tar 可以将很多档案『打包』成为一个档案！ 甚至是目录也可以这么玩。不过，单纯的 tar 功能仅是『打包』而已，亦即是将很多档案集结成为一个档案，事实上，他并没有提供压缩的功能，后来， GNU 计划中，将整个 tar 与压缩的功能结合在一起，如此一来提供使用者更方便并且更强大的压缩与打包功能！底下我们就来谈一谈这些在 Linux 底下基本的压缩指令吧！

---

compress

```
[root@linux ~]# compress [-dcr] 档案或目录
参数：
-d  ：用来解压缩的参数
-r  ：可以连同目录下的档案也同时给予压缩呢！
-c  ：将压缩数据输出成为 standard output (输出到屏幕)
范例：
范例一：将 /etc/man.config 复制到 /tmp ，并加以压缩
[root@linux ~]# cd /tmp
[root@linux tmp]# cp /etc/man.config .
[root@linux tmp]# compress man.config
[root@linux tmp]# ls -l
-rw-r--r--  1 root root 2605 Jul 27 11:43 man.config.Z

范例二：将刚刚的压缩档解开
[root@linux tmp]# compress -d man.config.Z

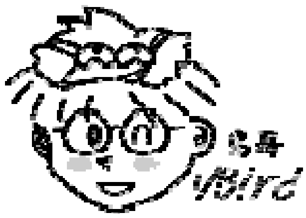
范例三：将 man.config 压缩成另外一个档案来备份
[root@linux tmp]# compress -c man.config > man.config.back.Z
[root@linux tmp]# ll man.config*
-rw-r--r--  1 root root 4506 Jul 27 11:43 man.config
-rw-r--r--  1 root root 2605 Jul 27 11:46 man.config.back.Z
# 这个 -c 的参数比较有趣！他会将压缩过程的数据输出到屏幕上，而不是写入成为
# file.Z 档案。所以，我们可以透过数据流重导向的方法将数据输出成为另一个档名。
# 关于数据流重导向，我们会在 bash shell 当中详细谈论的啦！
```

这是用来压缩与解压缩附档名为 *.Z 的指令！所以看到 *.Z 的档案时，就应该要知道他是经由 compress 这个程序压缩的呦！这是最简单的压缩指令啰！不过，使用的时候需要特别留意的是， 当你以 compress 压

缩之后，如果没有下达其它的参数，那么原本的档案就会被后来的 *.Z 所取代！以上面的案例来说明：原本压缩的档案为 man.config ，那么当压缩完成之后，将只会剩下 man.config.Z 这个经过压缩的档案啰！那么解压缩呢？呵呵，则是将 man.config.Z 解压缩成 man.config ！ 使用上很简单啦！解压缩除了可以使用 compress -d 这个参数之外，也可以直接使用 uncompress ！意思相同啦！

另外，如果不想让原本的档案被更名成为 *.Z ，而想制作出另外的一个档名时，就可以利用数据流重导向，亦即是那个大于 (>) 的符号，将原本应该在屏幕上面出现的数据给他储存到其它档案去。 当然，这要加上 -c 的参数才行～关于数据流重导向，我们会在第三篇提到的！此外， compress 已经很少人在使用了，因为这支程序无法解开 *.gz 的档案，而 gzip 则可以解决 *.Z 的档案， 所以，如果您的 distribution 上面没有 compress 的话，没有关系的喔！

Tips:
compress 使用的频率越来越低了，如果您还是想要练习这个指令的话， 在 FC4 里头，他是在 ucompress 这个套件名称的套件内。您可以参考 RPM 的方式来安装！



---

gzip, zcat

```
[root@linux ~]# gzip [-cdt#] 檔名
[root@linux ~]# zcat 檔名.gz
参数：
-c  ：将压缩的数据输出到屏幕上，可透过数据流重导向来处理；
-d  ：解压缩的参数；
-t  ：可以用来检验一个压缩档的一致性～看看档案有无错误；
-#  ：压缩等级，-1 最快，但是压缩比最差、-9 最慢，但是压缩比最好！预设是 -6 ～
范例：
范例一：将 /etc/man.config 复制到 /tmp ，并且以 gzip 压缩
[root@linux ~]# cd /tmp
[root@linux tmp]# cp /etc/man.config .
[root@linux tmp]# gzip man.config
# 此时 man.config 会变成 man.config.gz ！

范例二：将范例一的档案内容读出来！
[root@linux tmp]# zcat man.config.gz
# 此时屏幕上会显示 man.config.gz 解压缩之后的档案内容！！

范例三：将范例一的档案解压缩
[root@linux tmp]# gzip -d man.config.gz

范例四：将范例三解开的 man.config 用最佳的压缩比压缩，并保留原本的档案
[root@linux tmp]# gzip -9 -c man.config > man.config.gz
```

gzip 是用来压缩与解压缩附档名为 *.gz 的指令!所以看到 *.gz 的档案时,就应该要知道他是经由 gzip 这个程序压缩的呦！另外， gzip 也提供 压缩比的服务！ -1 是最差的压缩比，但是压缩速度最快，而 -9

虽然可以达到较佳的压缩比（经过压缩之后，档案比较小一些！），但是却会损失一些速度！预设是 -6 这个数值！ gzip 也是相当常使用的一个压缩指令呢！

至于 zcat 则是用来读取压缩文件数据内容的指令！假如我们刚刚压缩的档案是一个文字文件， 那么你还记得如何读取文字文件吗？！没错！就是使用 cat ，那么读取压缩档呢？呵呵！就是使用 zcat 啰！由于 gzip 这个压缩指令主要想要用来取代 compress 的，所以 compress 的压缩档案也可以使用 gzip 来解开喔！同时， zcat 这个指令可以同时读取 compress 与 gzip 的压缩档呦！

---

bzip2, bzcat

```
[root@linux ~]# bzip2 [-cdz] 檔名
[root@linux ~]# bzcat 檔名.bz2
参数：
-c  ：将压缩的过程产生的数据输出到屏幕上！
-d  ：解压缩的参数
-z  ：压缩的参数
-#  ：与 gzip 同样的，都是在计算压缩比的参数， -9 最佳， -1 最快！
范例：
范例一：将刚刚的 /tmp/man.config 以 bzip2 压缩
[root@linux tmp]# bzip2 -z man.config
# 此时 man.config 会变成 man.config.bz2 !

范例二：将范例一的档案内容读出来！
[root@linux tmp]# bzcat man.config.bz2
# 此时屏幕上会显示 man.config.bz2 解压缩之后的档案内容！！

范例三：将范例一的档案解压缩
[root@linux tmp]# bzip2 -d man.config.bz2

范例四：将范例三解开的 man.config 用最佳的压缩比压缩，并保留原本的档案
[root@linux tmp]# bzip2 -9 -c man.config > man.config.bz2
```

使用 compress 附档名自动建立为 .Z ，使用 gzip 附档名自动建立为 .gz 。这里的 bzip2 则是自动的将附档名建置为 .bz2 啰！所以当我们使用具有压缩功能的 bzip2 -z 时，那么刚刚的 man.config 就会自动的变成了 man.config.bz2 这个档名啰！

好了，那么如果我想要读取这个档案的内容呢？ 是否一定要解开？当然不需要啰！可以使用简便的 bzcat 这个指令来读取内容即可！例如上面的例子中， 我们可以使用 bzcat man.config.bz2 来读取数据而不需要解开！此外，当您要解开一个压缩档时， 这个档案的名称为 .bz, .bz2, .tbz, .tbz2 等等，那么就可以尝试使用 bzip2 来解看看啦！当然啰，也可以使用 bunzip2 这个指令来取代 bzip2 -d 啰。

---

tar

```
[root@linux ~]# tar [-cxtzjvfpPN] 档案与目录 ....
参数：
-c  ：建立一个压缩档案的参数指令(create 的意思)；
-x  ：解开一个压缩档案的参数指令！
-t  ：查看 tarfile 里面的档案！
      特别注意，在参数的下达中， c/x/t 仅能存在一个！不可同时存在！
      因为不可能同时压缩与解压缩。
-z  ：是否同时具有 gzip 的属性？亦即是否需要用 gzip 压缩？
-j  ：是否同时具有 bzip2 的属性？亦即是否需要用 bzip2 压缩？
-v  ：压缩的过程中显示档案！这个常用，但不建议用在背景执行过程！
-f  ：使用档名，请留意，在 f 之后要立即接档名喔！不要再加参数！
      例如使用『 tar -zcvfP tfile sfile』就是错误的写法，要写成
      『 tar -zcvPf tfile sfile』才对喔！
-p  ：使用原档案的原来属性（属性不会依据使用者而变）
-P  ：可以使用绝对路径来压缩！
-N  ：比后面接的日期(yyyy/mm/dd)还要新的才会被打包进新建的档案中！
--exclude FILE：在压缩的过程中，不要将 FILE 打包！
范例：
范例一：将整个 /etc 目录下的档案全部打包成为 /tmp/etc.tar
[root@linux ~]# tar -cvf /tmp/etc.tar /etc  <==仅打包，不压缩！
[root@linux ~]# tar -zcvf /tmp/etc.tar.gz /etc  <==打包后，以 gzip 压缩
[root@linux ~]# tar -jcvf /tmp/etc.tar.bz2 /etc  <==打包后，以 bzip2 压缩
# 特别注意，在参数 f 之后的档案档名是自己取的，我们习惯上都用 .tar 来作为辨识。
# 如果加 z 参数，则以 .tar.gz 或 .tgz 来代表 gzip 压缩过的 tar file ～
# 如果加 j 参数，则以 .tar.bz2 来作为附档名啊～
# 上述指令在执行的时候，会显示一个警告讯息：
# 『tar: Removing leading `/' from member names』那是关于绝对路径的特殊设定。

范例二：查阅上述 /tmp/etc.tar.gz 档案内有哪些档案？
[root@linux ~]# tar -ztvf /tmp/etc.tar.gz
# 由于我们使用 gzip 压缩，所以要查阅该 tar file 内的档案时，
# 就得要加上 z 这个参数了！这很重要的！

范例三：将 /tmp/etc.tar.gz 档案解压缩在 /usr/local/src 底下
[root@linux ~]# cd /usr/local/src
[root@linux src]# tar -zxvf /tmp/etc.tar.gz
# 在预设的情况下，我们可以将压缩档在任何地方解开的！以这个范例来说，
# 我先将工作目录变换到 /usr/local/src 底下，并且解开 /tmp/etc.tar.gz ，
# 则解开的目录会在 /usr/local/src/etc 呢！另外，如果您进入 /usr/local/src/etc
# 则会发现，该目录下的档案属性与 /etc/ 可能会有所不同喔！

范例四：在 /tmp 底下，我只想要将 /tmp/etc.tar.gz 内的 etc/passwd 解开而已
[root@linux ~]# cd /tmp
```

```
[root@linux tmp]# tar -zxvf /tmp/etc.tar.gz etc/passwd
# 我可以透过 tar -ztvf 来查阅 tarfile 内的文件名称，如果单只要一个档案，
# 就可以透过这个方式来下达！注意到！ etc.tar.gz 内的根目录 / 是被拿掉了！

范例五：将 /etc/ 内的所有档案备份下来，并且保存其权限！
[root@linux ~]# tar -zcvpf /tmp/etc.tar.gz /etc
# 这个 -p 的属性是很重要的，尤其是当您要保留原本档案的属性时！

范例六：在 /home 当中，比 2005/06/01 新的档案才备份
[root@linux ~]# tar -N '2005/06/01' -zcvf home.tar.gz /home

范例七：我要备份 /home, /etc ，但不要 /home/dmtsai
[root@linux ~]# tar --exclude /home/dmtsai -zcvf myfile.tar.gz /home/* /etc

范例八：将 /etc/ 打包后直接解开在 /tmp 底下，而不产生档案！
[root@linux ~]# cd /tmp
[root@linux tmp]# tar -cvf - /etc | tar -xvf -
# 这个动作有点像是 cp -r /etc /tmp 啦～依旧是有其有用途的！
# 要注意的地方在于输出档变成 - 而输入档也变成 - ，又有一个 | 存在～
# 这分别代表 standard output, standard input 与管线命令啦！
# 这部分我们会在 Bash shell 时，再次提到这个指令跟大家再解释啰！
```

这是一个多用途的压缩指令！刚刚我们提到的 compress 与 gzip 是可以适用在一个档案的压缩上面， 但是如果是要将一个目录压缩成一个档案呢？！这时该如何是好？！呵呵！ tar 就派上用场了！ tar 可以将整个目录或者是指定的档案都整合成一个档案！例如上面的范例一，他可以将 /etc 底下的档案全部整合成一个档案！同时， tar 可以配合 gzip（这个 gzip 的功能已经已经附加上 tar 里面去了），同时整合并压缩！呵呵！很方便吧！

『 tar 用来作备份是很重要的指令！ 』而由于 tar 整合过后的档案我们通常会取名为 *.tar ，而如果还含有 gzip 的压缩属性，那么就取名为 *.tar.gz 啰！取这个文件名只是为了方便我们记忆这个档案是什么属性罢了！并没有实际的意义在！

- 绝对路径与权限的问题

另外，需要注意的是，在使用的参数方面，有还有几个有用的参数需要来了解一番，亦即是 -p 与 -P 这两个！在我们的范例一当中，有提到一个警告讯息，那就是『 tar: Removing leading `/' from member names』意思是说， tar 将 /etc 目录的那个 / 拿掉了！这是因为担心未来你在解开压缩的时候，会产生一些困扰， 因为在 tar 里面的档案如果是具有『绝对路径』的话，那么你解开的档案将会『一定』在该路径下也就是 /etc，而不是相对路径（这里请用心的想一想！）。

这样子的最大困扰是，万一有人拿走了你的这个档案，并且将该档案在他的系统上面解开！万一他的系统上面正巧也有 /etc 这个目录（那当然是一定有的啊！），哈哈！他的档案就会『正巧』被覆盖了！ 所以啰，在预设的情况中，如果是以『绝对路径』来建立打包档案，那么 tar 将会自动的将 / 拿掉！这是为了刚刚说明的『安全』为前提所做的默认值。好了！但是你就是要以绝对路径来建立打包的档案！那么就

加入 -P 这个参数吧（请注意！是大写字符）！这样就可以啦！

那么 -p 是什么（小写字符）？呵呵！那个 -p 是 permission 的意思，也就是『权限』啦！使用 -p 之后，被打包的档案将不会依据使用者的身份来改变权限喔！

- 关于档案的更新日期：

这里还有一个值得注意的参数呦！那就是在备份的情况中很常使用的 -N 的这个参数！ 你可以参考一下上面的例子就可以知道啦！在这个例子当中，相当重要的就是那个日期啦！ 在备份的情况当中，我们都希望只要备份较新的档案就好了，为什么呢？因为旧的档案我们已经有备份啰！ 干嘛还要再备份一次，浪费时间也浪费系统资源！这个时候此一参数就显的相当的重要了啊！

- 关于 standard input/standard output：

在上面的例子中，最后一个例子很有趣『tar cvf - /etc | tar -xvf - 』！他是直接以管线命令『 pipe 』来进行压缩、解压缩的过程！在上面的例子中，我们想要『将 /etc 底下的资料直接 copy 到目前所在的路径，也就是 /tmp 底下来』，但是又觉得使用 cp -r 有点麻烦，那么就直接以这个打包的方式来打包，其中，指令里面的 - 就是表示那个被打包的档案啦！由于我们不想要让中间档案存在，所以就以这一个方式来进行复制的行为啦！

- 什么是 tarfile 与 tarball?

tar 的功能相当的多，而由于他是经由『打包』之后再处理的一个过程，所以常常我们会听到 tarball 的档案，那就是经由 tar 打包再压缩的档案啦！而如果仅是打包而没有压缩的话，我们就称为 tarfile 啰～此外， tar 也可以用在备份的储存媒体上面，最常见的就是磁带机了！假设我的磁带机代号为 /dev/st0 ，那么我要将我的 /home 底下的数据都给他备份上去时，就是使用 tar /dev/st0 /home 就可以啦！很不错吧！

在 Linux 当中， gzip 已经被整合在 tar 里面了！但是 Sun 或者其它较旧的 Unix 版本中， 当中的 tar 并没有整合 gzip ，所以如果你需要解压缩的话，就需要这么做：

```
gzip -d testing.tar.gz
tar -xvf testing.tar
```

第一个步骤会将档案解压缩，第二个步骤才是将数据解出来！与其它压缩程序不太一样的是， bzip2, gzip 与 compress 在没有加入特殊参数的时候，原先的档案会被取代掉，但是使用 tar 则原来的与后来的档案都会存在呦！

---

dd

我们在上一章当中，在制作出 swap file 时，使用过 dd 这个指令对吧？！ 不过，这个指令可不只是制作一个档案而已喔～这个 dd 指令最大的功效，鸟哥认为，应该是在于『备份』啊！ 因为 dd 可以读取装置的内容，然后将整个装置备份成一个档案呢！真的是相当的好用啊～ dd 的用途有很多啦～但是我们仅讲一些比较重要的参数，如下：

```
[root@linux ~]# dd if="input_file" of="outptu_file" bs="block_size" \
count="number"
参数：
if   ：就是 input file 啰～也可以是装置喔！
of   ：就是 output file 喔～也可以是装置；
```

```
bs   ：规划的一个 block 的大小，如果没有设定时，预设是 512 bytes
count：多少个 bs 的意思。
范例：

范例一：将 /etc/passwd 备份到 /tmp/passwd.back 当中
[root@linux ~]# dd if=/etc/passwd of=/tmp/passwd.back
3+1 records in
3+1 records out
[root@linux ~]# ll /etc/passwd /tmp/passwd.back
-rw-r--r--  1 root root 1746 Aug 25 14:16 /etc/passwd
-rw-r--r--  1 root root 1746 Aug 29 16:57 /tmp/passwd.back
# 仔细的看一下，我的 /etc/passwd 档案大小为 1746 bytes，因为我没有设定 bs ，
# 所以预设是 512 bytes 为一个单位，因此，上面那个 3+1 表示有 3 个完整的
# 512 bytes，以及未满 512 bytes 的另一个 block 的意思啦！
# 事实上，感觉好像是 cp 这个指令啦～

范例二：备份 /dev/hda 的 MBR
[root@linux ~]# dd if=/dev/hda of=/tmp/mbr.back bs=512 count=1
1+0 records in
1+0 records out
# 这就得好好了解一下啰～我们知道整颗硬盘的 MBR 为 512 bytes，
# 就是放在硬盘的第一个 sector 啦，因此，我可以利用这个方式来将
# MBR 内的所有数据都纪录下来，真的很厉害吧！ ^_^

范例三：将整个 /dev/hda1 partition 备份下来。
[root@linux ~]# dd if=/dev/hda1 of=/some/path/filenaem
# 这个指令很厉害啊！将整个 partition 的内容全部备份下来～
# 后面接的 of 必须要不是在 /dev/hda1 的目录内啊～否则，怎么读也读不完～
# 这个动作是很有效用的，如果改天你必须要完整的将整个 partition 的内容填回去，
# 则可以利用 dd if=/some/file of=/dev/hda1 来将数据写入到硬盘当中。
# 如果想要整个硬盘备份的话，就类似 Norton 的 ghost 软件一般，
# 由 disk 到 disk ，嘿嘿～利用 dd 就可以啦～厉害厉害！
```

你可以说， tar 可以用来备份关键数据，而 dd 则可以用来备份整颗 partition 或 整颗 disk ，很不错啊～不过，如果要将数据填回到 filesystem 当中， 可能需要考虑到原本的 filesystem 才能成功啊！

---

cpio

这个指令可有趣了！他是透过数据流重导向的方法来将档案进行输出/输入的一个方式～ 因为我们尚未提到数据流重导向，所以，您可以先略过这的指令的练习。等到后续的章节读完后， 再来这个章节瞧一瞧！

```
[root@linux ~]# cpio -covB  > [file|device] <==备份
[root@linux ~]# cpio -icduv < [file|device] <==还原
参数：
```

```
-o ：将数据 copy 输出到档案或装置上
-i ：将数据自档案或装置 copy 出来系统当中
-t ：查看 cpio 建立的档案或装置的内容
-c ：一种较新的 portable format 方式储存
-v ：让储存的过程中文件名称可以在屏幕上显示
-B ：让预设的 Blocks 可以增加至 5120 bytes ，预设是 512 bytes ！
     这样的好处是可以让大档案的储存速度加快(请参考 i-nodes 的观念)
-d ：自动建立目录！由于 cpio 的内容可能不是在同一个目录内，
     如此的话在反备份的过程会有问题！ 这个时候加上 -d 的话，
     就可以自动的将需要的目录建立起来了！
-u ：自动的将较新的档案覆盖较旧的档案！
范例：
范例一：将所有系统上的数据通通写入磁带机内！
[root@linux ~]# find / -print | cpio -covB > /dev/st0
# 一般来说，使用 SCSI 接口的磁带机，代号是 /dev/st0 喔！

范例二：检查磁带机上面有什么档案？
[root@linux ~]# cpio -icdvt < /dev/st0
[root@linux ~]# cpio -icdvt < /dev/st0 > /tmp/content
# 第一个动作当中，会将磁带机内的文件名列出到屏幕上面，而我们可以透过第二个动作，
# 将所有的文件名通通纪录到 /tmp/content 档案去！

范例三：将磁带上的数据还原回来～
[root@linux ~]# cpio -icduv < /dev/st0
# 一般来说，使用 SCSI 接口的磁带机，代号是 /dev/st0 喔！

范例四：将 /etc 底下的所有『档案』都备份到 /root/etc.cpio 中！
[root@linux ~]# find /etc -type f | cpio -o > /root/etc.cpio
# 这样就能够备份啰～您也可以将数据以 cpio -i < /root/etc.cpio
# 来将资料捉出来！！！！
```

这个 cpio 还蛮神奇的呢！他最适用于备份的时候使用的一个指令了！为什么呢？因为他并不像 cp 一样，可以直接的将档案给他 copy 过去，例如 cp * /tmp 就可以将所在目录的所有档案 copy 到 /tmp 底下，在 cpio 这个指令的用法中，由于 cpio 无法直接读取档案， 而是需要『每一个档案或目录的路径连同文件名一起』才可以被记录下来！因此， cpio 最常跟 find 这个指令一起使用了！

这个 cpio 好像不怎么好用呦！但是，嘿嘿！他可是备份的时候的一项利器呢！因为他可以备份任何的档案， 包括 /dev 底下的任何装置档案！呵呵！所以他可是相当重要的呢！！您说是吧！ 而由于 cpio 必需要配合其它的程序，例如 find 来建立档名，所以， cpio 与管线命令及数据流重导向的相关性就相当的重要了！

---

vi 文字处理器

最近更新日期：2005/08/01

每个系统管理员都应该至少要学会一种文字接口的文书处理器，以方便系统日常的管理行为。在 Linux 上头的文字处理软件非常的多，不过，鸟哥还是建议使用 vi 这个正规的文书处理器。 这是因为 vi 几乎在任何一个 Unix Like 的机器都存在，学会他，轻松很多啊！ 此外，后来 GNU 计划有推出 vim 这个 vi 的进阶版本，可以用的额外功能更多了！ vi 是未来我们进行 shell script 程序的编写与服务器设定的重要工具喔！ 而且是非常非常重要的工具，一定要学会才行啊！ ^_^

---

vi 与 vim

由前面一路走来，我们一直建议使用文字模式来处理 Linux 的系统设定问题，因为不但可以让您比较容易了解到 Linux 的运作状况，也比较容易了解整个设定的基本精神，更能『保证』您的修改可以顺利的被运作。 所以，在 Linux 的系统中使用文字编辑器来编辑您的 Linux 参数设定档，嗯！可是一件很重要的事情呦！所以说嘛！系统管理员至少应该要熟悉一种文书处理器的！

> Tips:
> 这里要再次的强调，不同的 Linux distribution 各有其不同的附加软件，例如 Red Hat 与 Fedora 的 userconf, Linuxconf, ntsysv 与 setup 等等，而 SuSE 则有 YOU 管理工具等等，因此，如果您只会使用此种类型的软件来控制您的 Linux 系统时，当接管不同的 Linux distributions 时，呵呵！那可就苦恼了！



由 Linux是什么 介绍中，我们知道 Linux 与 Unix 系统中的参数文件几乎都是 ASCII 码的『纯文字』文件！因此，利用简单的文字编辑软件就可以马上修改 Linux 的参数档啰！然而，与 Windows 不同的是，如果您用惯了 Microsoft Word 或 Corel Wordperfect 的话，那么除了 X window 里面的编辑程序(如 xemacs)用起来尚可应付外，于 Linux 的文字模式下，会觉得档案编辑程序都没有 Windows 程序那么方便。

Tips:
还记得什么是纯文字文件吗？忘记的话，回到 Linux 的档案属性与目录配置 里头去瞧一瞧先～该档案格式以 ASCII 格式码为主。说穿了，就是您『不论使用什么编辑器』来开启那个档案时，都可以将内容给您看到，而不是呈现乱码的档案， 那就是纯文字文件了！当您以 Windows 的 word 存一个档案时，在 DOS 的情况下使用 type 这个指令来查阅数据，嗯！完全不知到内容是什么？因为会出现很多的乱码，那并非是纯文字文件，而如果以 word 在存盘时，选择『纯文字类型』，嗯！那就可以使用 type 看到该档案的内容了！ 由于纯文字文件在任何操作系统底下都可以被取用，是相当方便的一种设定格式啊！



无论如何，要管理好 Linux 系统时，纯文字的手工设定仍是需要的！那么在 Linux 底下有哪些文书编辑器呢？ 可多了～例如 vi, emacs, xemacs, joe, e3, xedit, kedit, pico .... 多的很～ 各家处理器各有其优缺点，您当然可以选择任何一个您觉得适用的文书处理器来使用。不过，鸟哥还是比较建议使用 vi 啦！这是因为 vi 是 Unix Like 的机器上面预设都有安装的软件，也就是说，您一定可以接触到这个软件就是了。 另外，在较新的 distributions 上，您也可以使用较新较先进的 vim 这个文书处理器！ vim 可以看做是 vi 的进阶软件，他可以具有颜色显示，很方便程序开发人员在进行程序的撰写呢！

简单的来说， vi 是老式的文书处理器，不过功能已经很齐全了，但是还是有可以进步的地方。 vim 则可以说是程序开发者的一项很好用的工具，就连 vim 的官方网站 (http://www.vim.org) 自己也说 vim 是一个『程序开发工具』而不是文字处理软件～^_^。 因为 vim 里面加入了很多额外的功能，例如支持正规表示法的搜寻架构、多档案编辑、区块复制等等。 这对于我们在 Linux 上面进行一些设定档的修订工作时，是很棒的一项功能呢！

底下鸟哥会先就简单的 vi 做个介绍，然后再跟大家报告一下 vim 的额外功能与用法呢！

---

vi 的使用

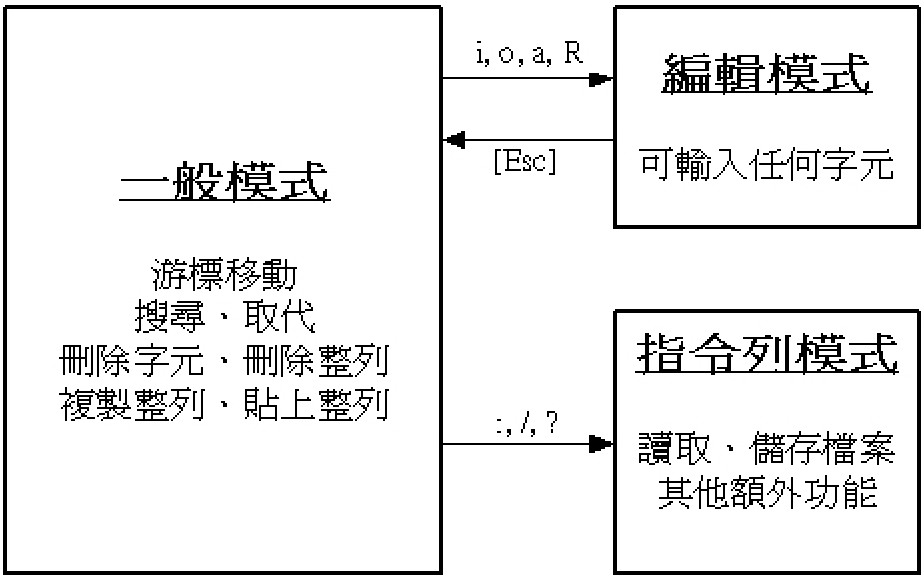
基本上 vi 共分为三种模式，分别是『一般模式』、 『编辑模式』与『指令列命令模式』三种！ 这三种模式的作用是：

- 一般模式：
  以 vi 处理一个档案的时后，一进入该档案就是一般模式了。在这个模式中， 你可以使用『上下左右』按键来移动光标，您可以使用『删除字符』或『删除整行』来处理档案内容， 也可以使用『复制、贴上』来处理您的文件数据。

- 编辑模式：
  在一般模式中可以处理删除、复制、贴上等等的动作，但是却无法编辑的！ 要等到您按下『i, I, o, O, a, A, r, R』等字母之后才会进入编辑模式。注意了！通常在 Linux 中，按下上述的字母时，在画面的左下方会出现『 INSERT 或 REPLACE 』的字样，才可以输入任何字来输入到您的档案中！而如果要回到一般模式时， 则必须要按下『Esc』这个按键即可退出编辑模式。
- 指令列命令模式：
  在一般模式当中，输入『 : 或 / 或 ? 』就可以将光标移动到最底下那一行，在这个模式当中，

可以提供您『搜寻资料』的动作，而读取、存盘、大量取代字符、离开 vi 、显示行号 等等的动作则是在此模式中达成的！

简单的说，我们可以将这三个模式想成底下的图标来表示之：



图一、 vi 三种模式的相互关系

闲话不多说，我们底下以一个简单的例子来进行说明吧！

简易执行范例

我们怎么使用 vi 建立一个档名为 test.txt 的资料呢？也是很简单的啦， 整个步骤可以是这样：

1. 使用 vi 进入一般模式；

```
[root@linux ~]# vi test.txt
```

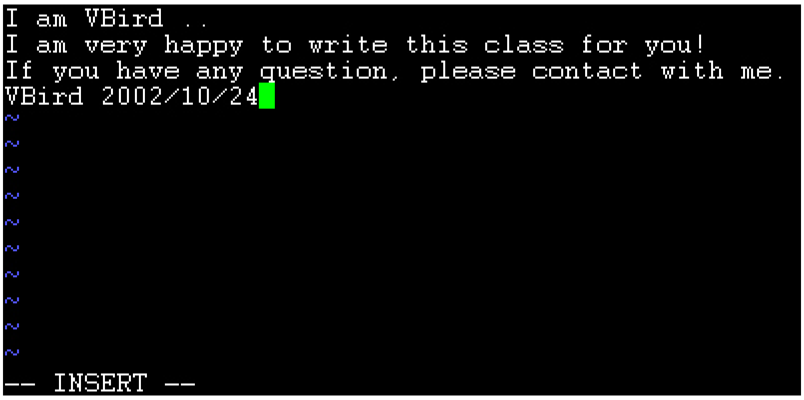
直接输入『 vi 档名』即可进入 vi 了！如下图所示，左下角还会显示这个档案目前的状态！如果是新建档案会显示 [New File]，如果是已存在的档案，则会显示目前的文件名、行数与字符数，例如：『"/etc/man.config" 145L, 4614C』



图二、 利用 vi 开启一个档案

2. 按下 i 进入编辑模式，开始编辑文字；

在一般模式之中，只要按下 I, o, a 等字符，就可以进入编辑模式了！在编辑模式当中，您可以发现在左下角会出现 – INSERT- 的画面，那就是可以输入任意字符的提示啰！这个时候，键盘上除了 [Esc] 这个按键之外，其它的按键都可以视作为一般的输入按钮了，所以您可以进行任何的编辑啰！（注：在 vi 里面， [tab] 这个按钮所得到的结果与空格符所得到的结果是不一样的，特别强调一下！）



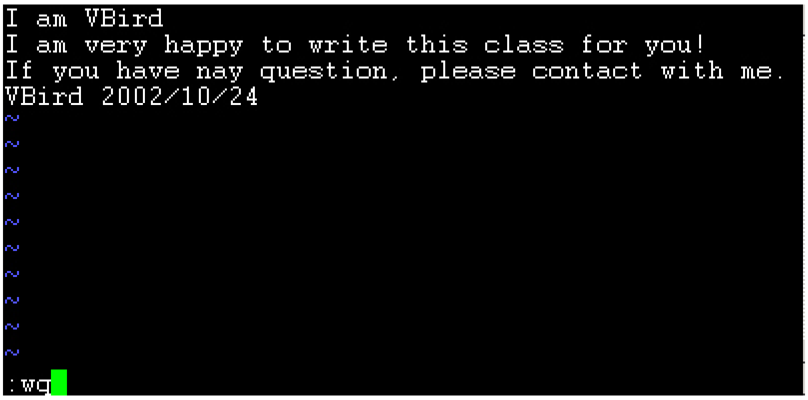
图三、 进入 vi 的编辑模式

3. 按下 [ESC] 按钮回到一般模式；

好了，假设我已经按照上面的样式给他编辑完毕了，那么应该要如何退出呢？是的！没错！就是给他按下 [Esc] 这个按钮即可！马上你就会发现画面左下角的 – INSERT – 不见了！

4. 在一般模式中按下 :wq 储存后离开 vi ！

OK，我们要存档了，存盘并离开的指令很简单，输入『:wq』即可存档离开！（注意了，按下 : 该光标就会移动到最底下一行去！）这时你在提示字符后面输入『 ls -l 』即可看到我们刚刚建立的 test.txt 档案啦！整个图示有点像底下这样：



图四、利用 vi 储存档案

如此一来，您的档案 test.txt 就已经建立起来啰！很简单吧！需要注意的是，如果您的档案权限不对，例如为 -r--r--r-- 时，那么可能会无法写入，那么可以使用『强制写入』的方式吗？可以！使用『:wq!』多加一个惊叹号即可！不过，需要特别注意呦！那个是在『您的权限可以改变』的情况下才能成立的！ 关于权限的概念，请参考一下 Linux 的档案权限概念 啰！

命令列内容说明

如前所述，所谓的命令列或命令模式，就是在最下面一行没有显示 『--INSERT--』或者『--REPLACE--』 字样的时候。通常在命令列中的指令有下面几种：（注意，当按下 『:』 时，光标会自动移动到屏幕的最下面一行！）

一般模式： 移动光标的方法	
h 或 向左方向键(←)	光标向左移动一个字符
j 或 向下方向键(↓)	光标向下移动一个字符
k 或 向上方向键(↑)	光标向上移动一个字符
l 或 向右方向键(→)	光标向右移动一个字符
如果想要进行多次移动的话，例如向下移动 30 行，可以使用 "30j" 或 "30↓" 的组合按键， 亦即加上想要进行的次数(数字)后，按下动作即可！	
[Ctrl] + [f]	屏幕『向下』移动一页，相当于 [Page Down]按键 (常用)
[Ctrl] + [b]	屏幕『向上』移动一页，相当于 [Page Up] 按键 (常用)
[Ctrl] + [d]	屏幕『向下』移动半页

[Ctrl] + [u]	屏幕『向上』移动半页
+	光标移动到非空格符的下一列
-	光标移动到非空格符的上一列
n<space>	那个 n 表示『数字』，例如 20 。按下数字后再按空格键，光标会向右移动这一行的 n 个字符。例如 20<space> 则光标会向后面移动 20 个字符距离。
0	这是数字『 0 』：移动到这一行的最前面字符处 (常用)
$	移动到这一行的最后面字符处(常用)
H	光标移动到这个屏幕的最上方那一行
M	光标移动到这个屏幕的中央那一行
L	光标移动到这个屏幕的最下方那一行
G	移动到这个档案的最后一行(常用)
nG	n 为数字。移动到这个档案的第 n 行。例如 20G 则会移动到这个档案的第 20 行(可配合 :set nu)
gg	移动到这个档案的第一行，相当于 1G 啊！ (常用)
n<Enter>	n 为数字。光标向下移动 n 行(常用)
一般模式： 搜寻与取代	
/word	向光标之下寻找一个字符串名称为 word 的字符串。例如要在档案内搜寻 vbird 这个字符串，就输入 /vbird 即可！ (常用)
?word	向光标之上寻找一个字符串名称为 word 的字符串。
n	这个 n 是英文按键。代表『重复前一个搜寻的动作』的意思。举例来说， 如果刚刚我们执行 /vbird 去向下搜寻 vbird 这个字符串，则按下 n 后，会向下继续搜寻下一个名称为 vbird 的字符串。如果是执行 ?vbird 的话，那么按下 n 则会向上继续搜寻名称为 vbird 的字符串！
N	这个 N 是英文按键。与 n 刚好相反，为『反向』进行前一个搜寻动作。 例如 /vbird 后，按下 N 则表示『向上』搜寻 vbird 。
:n1,n2s/word1/word2/g	n1 与 n2 为数字。在第 n1 与 n2 行之间寻找 word1 这个字符串，并将该字符串取代为 word2 ！举例来说，在 100 到 200 行之间搜寻 vbird 并取代为 VBIRD 则：『:100,200s/vbird/VBIRD/g』。(常用)
:1,$s/word1/word2/g	从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2 ！(常用)
:1,$s/word1/word2/gc	从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2 ！且在取代前显示提示字符给使用者确认 (conform) 是否需要取代！(常用)

一般模式： 删除、复制与贴上	
x, X	在一行字当中，x 为向后删除一个字符 (相当于 [del] 按键)， X 为向前删除一个字符(相当于 [backspace] 亦即是退格键) (常用)
nx	n 为数字，连续向后删除 n 个字符。举例来说，我要连续删除 10 个字符， 『10x』。
dd	删除游标所在的那一整列(常用)
ndd	n 为数字。删除光标所在的向下 n 列，例如 20dd 则是删除 20 列 (常用)
d1G	删除光标所在到第一行的所有数据
dG	删除光标所在到最后一行的所有数据
d$	删除游标所在处，到该行的最后一个字符
d0	那个是数字的 0 ，删除游标所在处，到该行的最前面一个字符
yy	复制游标所在的那一行(常用)
nyy	n 为数字。复制光标所在的向下 n 列，例如 20yy 则是复制 20 列 (常用)
y1G	复制光标所在列到第一列的所有数据
yG	复制光标所在列到最后一列的所有数据
y0	复制光标所在的那个字符到该行行首的所有数据
y$	复制光标所在的那个字符到该行行尾的所有数据
p, P	p 为将已复制的数据在光标下一行贴上，P 则为贴在游标上一行！ 举例来说，我目前光标在第 20 行，且已经复制了 10 行数据。则按下 p 后， 那 10 行数据会贴在原本的 20 行之后，亦即由 21 行开始贴。但如果是按下 P 呢？ 那么原本的第 20 行会被推到变成 30 行。 (常用)
J	将光标所在列与下一列的数据结合成同一列
c	重复删除多个数据，例如向下删除 10 行，[10cj]
u	复原前一个动作。(常用)
[Ctrl]+r	重做上一个动作。(常用)
这个 u 与 [Ctrl]+r 是很常用的指令！一个是复原，另一个则是重做一次～ 利用这两个功能按键，您的编辑，嘿嘿！很快乐的啦！	
.	不要怀疑！这就是小数点！意思是重复前一个动作的意思。 如果您想要重复删除、重复贴上等等动作，按下小数点『.』就好了！ (常用)
进入编辑模式	
i, I	插入：在目前的光标所在处插入输入之文字，已存在的文字会向后

	退； 其中， i 为『从目前光标所在处插入』， I 为『在目前所在行的第一个非空格符处开始插入』。 (常用)
a, A	a 为『从目前光标所在的下一个字符处开始插入』， A 为『从光标所在行的最后一个字符处开始插入』。(常用)
o, O	这是英文字母 o 的大小写。o 为『在目前光标所在的下一行处插入新的一行』；O 为在目前光标所在处的上一行插入新的一行！(常用)
r, R	取代：r 会取代光标所在的那一个字符；R 会一直取代光标所在的文字，直到按下 ESC 为止；(常用)
上面这些按键中，在 vi 画面的左下角处会出现『--INSERT--』或『--REPLACE--』的字样。 由名称就知道该动作了吧！！特别注意的是，我们上面也提过了，你想要在档案里面输入字符时， 一定要在左下角处看到 INSERT/REPLACE 才能输入喔！	
Esc	退出编辑模式，回到一般模式中(常用)
指令列命令模式	
:w	将编辑的数据写入硬盘档案中(常用)
:w!	若档案属性为『只读』时，强制写入该档案。不过，到底能不能写入， 还是跟您对该档案的档案权限有关啊！
:q	离开 vi (常用)
:q!	若曾修改过档案，又不想储存，使用 ! 为强制离开不储存档案。
注意一下啊，那个惊叹号 (!) 在 vi 当中，常常具有『强制』的意思～	
:wq	储存后离开，若为 :wq! 则为强制储存后离开 (常用)
:e!	将档案还原到最原始的状态！
ZZ	若档案没有更动，则不储存离开，若档案已经经过更动，则储存后离开！
:w [filename]	将编辑的数据储存成另一个档案（类似另存新档）
:r [filename]	在编辑的数据中，读入另一个档案的数据。亦即将 『filename』 这个档案内容加到游标所在行后面
:n1,n2 w [filename]	将 n1 到 n2 的内容储存成 filename 这个档案。
:! command	暂时离开 vi 到指令列模式下执行 command 的显示结果！例如『:! ls /home』即可在 vi 当中察看 /home 底下以 ls 输出的档案信息！
:set nu	显示行号，设定之后，会在每一行的前缀显示该行的行号
:set nonu	与 set nu 相反，为取消行号！

特别注意，在 vi 中，『数字』是很有意义的！数字通常代表重复做几次的意思！ 也有可能是代表去到第几个什么什么的意思。举例来说，要删除 50 行，则是用 『50dd』 对吧！ 数字加在动作之前～那我要向

下移动 20 行呢？那就是『20j』或者是『20↓』即可。

OK！会这些指令就已经很厉害了，因为常用到的指令也只有不到一半！通常 vi 的指令除了上面鸟哥注明的常用的几个外，其它是不用背的，你可以做一张简单的指令表在你的屏幕墙上， 一有疑问可以马上的查询呦！

一个案例练习

来来来！测试一下您是否已经熟悉 vi 这个指令呢？请依照底下的需求进行您的指令动作。 (底下的操作为使用 FC4 的预设档案来进行练习的。您可以在这里下载：
http://linux.vbird.org/linux_basic/0310vi/man.config。) 看看您的显示结果与鸟哥的结果是否相同啊？！

1. 请在 /tmp 这个目录下建立一个名为 vitest 的目录；
2. 进入 vitest 这个目录当中；
3. 将 /etc/man.config 拷贝到本目录底下(或由上述的连结下载 man.config 档案)；
4. 使用 vi 开启本目录下的 man.config 这个档案；
5. 在 vi 中设定一下行号；
6. 移动到第 58 行，向右移动 40 个字符，请问您看到的双引号内是什么目录？
7. 移动到第一行，并且向下搜寻一下『 bzip2 』这个字符串，请问他在第几行？
8. 接着下来，我要将 50 到 100 行之间的 man 改为 MAN，并且一个一个挑选是否需要修改，如何下达指令？
9. 修改完之后，突然反悔了，要全部复原，有哪些方法？
10. 我要复制 51 到 60 行这十行的内容，并且贴到最后一行之后；
11. 删除 11 到 30 行之间的 20 行；
12. 将这个档案另存成一个 man.test.config 的檔名；
13. 去到第 29 行，并且删除 15 个字符；
14. 储存后离开吧！

整个步骤可以如下显示：

1. 『mkdir /tmp/vitest』
2. 『cd /tmp/vitest』
3. 『cp /etc/man.config .』
4. 『vi man.config』
5. 『:set nu』
6. 先按下『58G』再按下『40→』会看到『/dir/bin/foo』这个字样在双引号内；
7. 先执行『1G』或『gg』后，直接输入『/bzip2』，则会去到第 116 行才对！
8. 直接下达『 :50,100s/man/MAN/gc 』即可！
9. (1)简单的方法可以一直按『 u 』回复到原始状态，(2)使用不储存离开『 :q! 』之后，再重新读取一次该档案；
10. 『51G』 然后再『 10yy 』之后按下『 G 』到最后一行，再给他『 p 』贴上十行！
11. 『11G』 之后，再给他『 20dd 』即可删除 20 行了；
12. :w man.test.config

13. 『29G』 之后，再给他『 15x 』即可删除 15 个字符；
14. 『:wq!』

如果您的结果都可以查的到，那么 vi 的使用上面应该没有太大的问题啦！剩下的问题会是在…..打字练习…。

关于档案的回复与暂存盘

另外，其实 vi 是具有『可回复』功能的呢！那么 vi 凭什么帮我们进行回复的功能呢？ 很简单啊！凭暂存档啊！举例来说，当我们编辑一个档案时，假设名称为 /tmp/passwd ， 那么在这个 /tmp 底下就会有一个临时文件，档名为『 /tmp/.passwd.swp 』，这是一个隐藏档， 我们所进行的一些修改都会暂时存在这个档案当中，万一在档案修改过程中， 系统挂了，那么下次你再重新 vi /tmp/passwd 时，系统就会告诉您，是否需要回复『Recovery』成修改过程中的模样？ 如果您按下 (R) ，嘿嘿！就可以将数据回复到修改过程的样子，而不是源文件啰！ 这是个很有用的功能喔！ ^_^

这也就是说，如果有一天，您去 /tmp 底下，执行 ls -al 时，发现到底下有两个档案， 档名分别为 passwd 与 .passwd.swp 的话，那么 (1) 可能有人在编辑这个档案； (2) 之前您在编辑这个档案时，因为某些不知名的因素导致 vi 程序中断，则该暂存档就会存在。 如果是状态 (2) ，则此时您可以将该 .passwd.swp 档案删除，或者是，直接 vi /tmp/passwd ， 在 vi 出现是否回复时，选择回复，然后储存更新 /tmp/passwd ，之后再将这个 /tmp/.passwd.swp 档案删除即可！ 如果不删除的话，那么每次编辑这个档案，都会告知您该档案有问题啊！ @_@

---

vim 的额外功能

其实，目前大部分的 distributions 都以 vim 取代 vi 的功能了！ 如果您使用 vi 后，却看到画面的右下角有显示目前光标所在的行列号码， 那么您的 vi 已经被 vim 所取代啰～ 为什么要用 vim 呢？因为 vim 具有颜色显示的功能，并且还支持许多的程序语法 (syntax)， 因此，当您使用 vim 编辑程序时(不论是 C 语言，还是 shell script)， 我们的 vim 将可帮您直接进行『程序除错 (debug)』的功能！真的很不赖吧！ ^_^

如果您在文字模式下，输入 alias 时，出现这样的画面：

```
[root@linux ~]# alias
alias vi='vim'
```

这表示当您使用 vi 这个指令时，其实就是执行 vim 啦！如果您没有这一行，那么您就必须要使用 vim filename 来启动 vim 啰！基本上， vim 的一般用法与 vi 完全一模一样～没有不同啦！那么我们就来看看 vim 的画面是怎样啰！假设我想要编辑 /etc/man.config ，则输入『vim /etc/man.config』

```
█
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.5p
```

```
#
# For more information about this file, see the man pages man(1)
# and man.conf(5).
"man.config" 138L, 4506C                              1,1           Top
```

上面的图示是 vim 一画面一角～他有几个特色要讲：

1. 最底下一行说明这个档案的特色，包括 138 行，共 4506 字符等等。
2. 那个 1,1 代表目前光标在第一行的第一个字符上。您可以看到第一行有个光标的存在啊！
3. 那个 Top 则表示，这个画面是整个档案的最上方！

至少就有这些信息。而在您移动光标时，那个 1,1 的游标定位也会跟着变动，是否很方便啊！ 好了，底下我们就来谈一谈其它 vim 的用法吧！

区块选择(Visual Block)

刚刚我们提到的简单的 vi 操作过程中，几乎提到的都是以行为单位的操作。那么如果我想要搞定的是一个区块范围呢？ 举例来说，像底下这种格式的档案：

```
192.168.1.1     host1.class.net
192.168.1.2     host2.class.net
192.168.1.3     host3.class.net
192.168.1.4     host4.class.net
.....中间省略......
```

这个档案我将他放置到 http://linux.vbird.org/linux_basic/0310vi/hosts ，您可以自行下载来看一看这个档案啊！如果我想要复制的只是前面的 IP 数字部分，后面的主机名称部分就不给他复制，那怎么办？这个时候就得需要使用区块选择(Visual Block)的功能了。当我们按下 v 或者 V 或者 [Ctrl]+v 时， 这个时候光标移动过的地方就会开始反白，这三个按键的意义分别是：

区块选择的按键意义	
v	字符选择，会将光标经过的地方反白选择！
V	行选择，会将光标经过的行反白选择！
[Ctrl]+v	区块选择，可以用长方形的方式选择资料
y	将反白的地方复制起来
d	将反白的地方删除掉

我们以上面的 IP 对应主机名称为范例，如果想要复制的是 IP 的话，而且仅想要前面四行， 那么我可以：

1. 将光标移动到第一行的第一个字符 （ 1G ）；
2. 然后按下 [Ctrl]+v （按着 [ctrl] 不放，再按下 v ）；
3. 然后移动方向键，向下向右移动数格，让整个反白区域涵盖 191.168.1.1 到 192.168.1.4 ；
4. 按下 y 复制 （此时反白会自动的不见）；

5. 移动到任何想要插入的区域，按下 p 就可以插入刚刚复制的区块内容！举例来说，移动到第 1 行的第 13 个字符处按下小写的 p ，看看会怎样？

这个区块选择在已经格式的档案中，就会显的很有帮助喔！尤其是我们想要大量复制其中一个区块， 而不是整行复制的场合中，就会很有用的啦！

多档案编辑

假设一个例子，你想要将刚刚我们的 hosts 内的 IP 复制到您的 /etc/hosts 这个档案去， 那么该如何编辑？我们知道在 vi 内可以使用 :r filename 来读入某个档案的内容， 不过，这样毕竟是将整个档案读入啊！如果我只是想要部分内容呢？呵呵！这个时候多档案同时编辑就很有用了。 我们可以使用 vim 后面同时接好几个档案来同时开启喔！ 相关的按键有：

多档案编辑的按键	
:n	编辑下一个档案
:N	编辑上一个档案
:files	列出目前这个 vim 的开启的所有档案

这个功能也很棒啊！现在您可以做一下练习看看说！假设您要将刚刚鸟哥提供的 hosts 内的 IP 复制到您的 /etc/hosts 档案内，那可以怎么进行呢？可以这样啊：

```
[root@linux ~]# vi hosts /etc/hosts
# 在这个档案中利用上个小节提到的区块选择，按下 [ctrl]+v 来进行区块选择，并复制。
# 然后按下 :n 在指令列的地方输入这玩意儿，就会转到下一个档案去，这个时候，
# 就可以按下 p 将刚刚复制的 IP 给贴到您的档案中啰！如果您按下 :files ，则：

--------------------------------------------------------------------
192.168.1.4     host4.class.net
192.168.1.5     host5.class.net
~
~
:files
  1 %a   "hosts"                    line 1
  2 #    "/etc/hosts"               line 1
Hit ENTER or type command to continue
--------------------------------------------------------------------

# 看到否？在指令列输入 :files 就可以显示目前所编辑的档案信息啰！
```

看到了吧？利用多档案编辑的功能，可以让您很快速的就将需要的资料复制到正确的档案内。 当然啰，这个功能也可以利用窗口接口来达到，那就是底下要提到的多窗口功能。

多窗口功能

想象两个情况：

- 当我有一个档案非常的大，我查阅到后面的数据时，想要『对照』前面的数据，是否需要使用 [ctrl]＋f 与 [ctrl]+b 来跑前跑后查阅？
- 我有两个需要对照着看的档案，不想使用前一小节提到的多档案编辑功能；

这样的情况下，开一个 vim 里头有两个窗口的环境，就有需要啦！那么如何开启新窗口呢？ 很简单啊！在指令列模式输入：『:sp {filename}』，那个 filename 可有可无， 如果想要在新窗口启动另一个档案，就加入档名，否则仅输入 :sp 时， 出现的则是同一个档案在两个窗口间！例如鸟哥使用 vim hosts 后，再以 :sp /etc/hosts ， 结果出现如下图示：

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain localhost
192.168.1.11    vbird-work
192.168.1.2     vbird-server
~

192.168.1.1    host1.class.net
192.168.1.2    host2.class.net
192.168.1.3    host3.class.net
192.168.1.4    host4.class.net
192.168.1.5    host5.class.net
```

怎样？帅吧！两个档案同时在一个屏幕上面显示，您还可以利用『[ctrl]+w+j』及『[ctrl]+w+k』 在两个窗口之间移动呢！这样的话，复制啊、查阅啊等等的，就变的很简单啰～ 指令的功能有很多，不过，您只要记得这几个就好了：

多窗口情况下的按键功能	
:sp [filename]	开启一个新窗口，如果有加 filename， 表示在新窗口开启一个新档案，否则表示两个窗口为同一个档案内容(同步显示)。
[ctrl]+wj	按键的按法是：先按下 [ctrl] 不放， 再按下 w 后放开所有的按键，然后再按下 j ，则光标可移动到下方的窗口。
[ctrl]+wk	同上，不过光标移动到上面的窗口。
[ctrl]+wq	其实就是 :q 结束离开啦！ 举例来说，如果我想要结束下方的窗口，那么利用 [ctrl]+wj 移动到下方窗口后，按下 :q 即可离开， 也可以按下 [ctrl]+wq 啊！

vim 环境设定

有没有发现，如果我们以 vim 软件来搜寻一个档案内部的某个字符串时，这个字符串会被反白， 而下次我们再次以 vim 编辑这个档案时，该搜寻的字符串还是存在呢！甚至于， 编辑其它档案时，如果其它档案也存在这个字符串，哇！！竟然还是主动反白耶！ 另外，当我们重复编辑同一个档案时，当第二次进入该档案时， 光标竟然就在上次离开的那一行上头呢！真是好方便啊～但是，怎么会这样呢？

这是因为我们的 vim 会主动的将您曾经做过的行为登录下来，好让您下次可以轻松的作业啊！ 那个记录动作的档案就是： ~/.viminfo 这个档案啦！每个人的家目录都应该会存在这个档案才对～这个档案是自动产生的， 您不必自行建立。而你在 vim 里头所做过的动作，就可以在这个档案内部查询到啰～ ^_^

此外，某些 distributions 的 vim 当中，利用搜寻时，他并不会显示反白， 有些 distributions 则会主动的帮您进行缩排的行为 (所谓的缩排， 就是当您按下 Enter 编辑新的一行时，光标不会在行首，而是在与上一行的第一个非空格符处对齐!)。这些其实都可以进行设定的,那就是 vim 的环境设定啰～ vim 的环境设定参数有很多，如果您想要知道目前的设定值，可以在一般模式时输入：『 :set all 』 来查阅，不过.....设定项目实在太多了～所以，鸟哥在这里仅列出一些平时比较常用的一些简单的设定值， 提供给您参考啊：

vim 的环境设定参数	
:set nu	还记得这个吧？！就是设定行号啊！取消的话，就是 :set nonu
:set hlsearch	这个就是设定是否将搜寻的字符串反白的设定值。 默认值就是 hlsearch ，如果不想要反白，就 :set nohlsearch 。
:set autoindent	是否自动缩排？autoindent 就是自动缩排， 不想要缩排就 :set noautoindent 。
:set backup	是否自动储存备份档？一般是 nobackup 的， 如果设定 backup 的话，那么当你更动任何一个档案时，则源文件会被另存成一个档名为 filename~ 的档案。 举例来说，我们编辑 hosts ，设定 :set backup ，那么当更动 hosts 时，在同目录下，就会产生 hosts~ 文件名的档案，记录原始的 hosts 档案内容～
:set ruler	还记得我们提到的右下角的一些状态列说明吗？ 这个 ruler 就是在显示或不显示该设定值的啦！
:set showmode	这个则是，是否要显示 --INSERT-- 之类的字眼在左下角的状态列。
:set backspace=(012)	一般来说， 如果我们按下 i 进入编辑模式后，可以利用退格键 (backspace) 来删除任意字符的。 但是，某些 distribution 则不许如此。此时，我们就可以透过 backspace 来设定啰～ 当 backspace 为 2 时，就是可以删除任意值；0 或 1 时，仅可删除刚刚输入的字符， 而无法删除原本就已经存在的文字了！
:set all	显示目前所有的环境参数设定值。
:syntax (off|on)	是否依据程序相关语法显示不同颜色？ 举例来说，在编辑一个纯文字文件时，如果开头是以 # 开始，那么该行就会变成蓝色。 如果您懂得写程序，那么这个 :syntax on 还会主动的帮您除错呢！但是，

	如果您仅是编写纯文本文件，要避免颜色对您的屏幕产生的干扰，则可以取消这个设定 :syntax off 。

总之，这些设定值很有用处的啦！但是......我是否每次使用 vim 都要重新设定一次各个参数值？ 这不太合理吧？！没错啊！所以，我们可以透过设定档来直接规定我们习惯的 vim 操作环境呢！ 整体 vim 的设定值一般是放置在 /etc/vimrc 这个档案，不过，不建议您修改他！你可以修改 ~/.vimrc 这个档案 (预设不存在，请您自行手动建立！)，将您所希望的设定值写入！ 举例来说，可以是这样的一个档案：

```
[root@linux ~]# vi ~/.vimrc
:set hlsearch
:set backspace=2
:set autoindent
:set ruler
:set showmode
:syntax on
```

这样，当您下次重新以 vim 编辑某个档案时，该档案的预设环境设定就是上头写的啰～ 这样，是否很方便您的操作啊！多多利用 vim 的环境设定功能呢！^_^

利用 vi 编辑前面章节的练习

我们前面提到很多的数据，例如 man 与 updatedb 等等的内容，对于设定档都是简单的提过而已。 在这里，我们就透过 vi 的编辑功能，来直接对我们前面提到的章节来进行一些练习吧！OK！来啰～

例题一：我今天自己安装了一套软件，这套软件的 man page 放置在 /opt/vbirdsoft/man 这个目录下，那我希望未来只要输入类似 man vbirdcommand 就可以查阅到我这个软件的说明文件，该如何是好？

答：

我以 FC4 为例， FC4 的 man page 设定档在 /etc/man.config 底下我可以找到该档案大约 47 行的地方，新增如下的数据：

```
MANPATH    /opt/vbirdsoft/man
```

储存后离开，从此以后，就可以查询到属于我自己的指令的在线说明文件了。

例题二：我知道查询档案可以利用 locate 来进行查询，但是，该程序必须要配合数据库的更新才行 (updatedb)。现在，我想让我的 FC4 每天进行档案数据库的更新，并且『不要更新 /var/cache 』这个目录，该如何是好？

答：

以 FC4 为例，他的 updatedb 数据库更新设定文件在 /etc/updatedb.conf 这个档案中。我以 vi 开启这个档案后将该档案修订成为：

```
DAILY_UPDATE=yes
PRUNEFS="selinuxfs usbdevfs NFS nfs afs sfs smbfs cifs autofs auto iso9660
```

udf"
PRUNEPATHS="/tmp /usr/tmp /var/spool/cups /var/spool/squid /var/tmp /afs /net /sfs
/selinux /udev /media /var/cache"

上列粗体部分为新加入的部分。那个 DAILY_UPDATE=yes 代表每日进行更新，至于 PRUNEPATHS 后面则接『不要更新的目录』，所以，这样就能够达到我们的需求啰！

例题三：在 partition 格式化的那个章节中，假设我有一个 partition 为 /dev/hdb5 ，这个 partition 挂载到 /disk2 上面，且目录 /disk2 已经建立好了。该 partition 使用的 filesystem 为 ext3 ，请问，如果我想要在开机的时候就挂载这个 partition ，该怎么办？
答：

开机挂载可以修改 /etc/fstab 这个档案，我在这个档案新增如下信息即可：

/dev/hdb5 /disk2 ext3 defaults 2 2

这样修改完毕后，下达 mount -a 测试看看能否正确挂载，之后就可以开机自动挂载啰～

vi 很重要的啦！上面的设定档都与 vi 编辑有关呢！重要重要喔！ ^_^

DOS 与 Linux 的断行字符

我们在 Linux 档案与目录管理 谈到 cat 时，曾经提到过 DOS 与 Linux 断行字符的不同。 而我们也可以利用 cat -A 来观察以 DOS (Windows 系统) 建立的档案的特殊格式， 也可以发现在 DOS 使用的断行字符为 ^M$ ，我们称为 CR 与 LF 两个符号。 而在 Linux 底下，则是仅有 LF ($) 这个断行符号。这个断行符号对于 Linux 的影响很大喔！ 为什么呢？

我们说过，在 Linux 底下的指令在开始执行时，他的判断依据是 『Enter』，而 Linux 的 Enter 为 LF 符号， 不过，由于 DOS 的断行符号是 CRLF ，也就是多了一个 ^M 的符号出来， 在这样的情况下，如果是一个 shell script 的程序档案，呵呵～将可能造成『程序无法执行』的状态～ 因为他会误判程序所下达的指令内容啊！这很伤脑筋吧！

那怎么办啊？很简单啊，将格式转换成为 Linux 即可啊！『废话』，这当然大家都知道，但是， 要以 vi 进入该档案，然后一个一个删除 CR 吗？当然没有这么没人性啦！ 我们可以透过简单的指令来进行格式的转换啊！

```
[root@linux ~]# dos2unix [-kn] file [newfile]
[root@linux ~]# unix2dos [-kn] file [newfile]
参数：
-k  ：保留该档案原本的 mtime 时间格式 (不更新档案上次内容经过修订的时间)
-n  ：保留原本的旧档，将转换后的内容输出到新档案，如： dos2unix -n old new
```

```
范例：
范例一：将我们提供的 hosts 档案格式更新为 dos 格式。
[root@linux ~]# unix2dos -k hosts
unix2dos: converting file hosts to DOS format ...
# 此时 hosts 这个档案的时间不会改变，但是内容主要将断行字符修改成为 DOS 的 CRLF 了。

范例二：将范例一已经变成 DOS 格式的 hosts 改名成为 hosts.dos ，并且转换 Linux
        格式到 hosts.linux
[root@linux ~]# mv hosts hosts.dos
[root@linux ~]# dos2unix -k -n hosts.dos hosts.linux
dos2unix: converting file hosts.dos to file hosts.linux in UNIX format ...
[root@linux ~]# ll
-rw-r--r--  1 root root      288 Aug  1 13:30 hosts.dos
-rw-------  1 root root      279 Aug  1 13:30 hosts.linux
# 嘿嘿！由于 DOS 格式当中多了 CR 字符，所以，档案比较大的啦！
```

因为断行字符以及 DOS 与 Linux 操作系统底下一些字符的定义不同，因此， 不建议您在 Windows 系统当中将档案编辑好之后，才上传到 Linux 系统，会容易发生错误问题。 而且，如果您在不同的系统之间复制一些纯文本文件时，千万记得要使用 unix2dos 或 dos2unix 来转换一下格式啊！

本章与 LPI 的关系：

在 http://www.lpi.org 所提供的 topic 当中， LPI 101 的 Topic 103 之 1.103.8 提到：『应试者应该能够学会使用 vi 文书编辑器，学习的目标包含了 vi 的插入、编辑、删除、复制与搜寻等功能的练习！』至于使用到的指令与 vi 当中所会用到的数据与指令为：

- vi
- /, ? (请看搜寻部分)
- h, j, k, l (移动光标的部分，这个较常考！)
- G, H, L (移动光标的部分)
- i, c, d, dd, p, o, a (删除、编辑与贴上的部分)
- ZZ, :w!, :q!, :e!
- :!

本章习题练习：

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 我要在某个档案的第 34 行向右移动 15 个字符，应该在一般模式下达什么指令？
- 先按下 34G 到第 24 行；
- 再按下 [15 + 向右键]，或 [15l] 亦可！

- 在 vi 里面， PageDown 按钮可以使用什么组合键来取代？

[Ctrl] + f 可以向后翻一页

- 如何去到 vi 该档案里面的页首或页尾?

去页首按下 1G ; 去页尾按下 G 即可

- 如何在一行中，移动到行头及行尾?

移动到行头，按 0 ，移动到行尾按 $ 即可!

- vi 里面， r 有什么功能?

取代光标所在的那个字符

- 如何将目前的页面另存新档?

:w filename

- 在 linux 底下最常使用的文书编辑器为 vi ，请问如何进入编辑模式?

在一般模式底下输入: i, I, a, A 为在本行当中输入新字符;(出现 –Insert-)

在一般模式当中输入: o, O 为在一个新的一行输入新字符;

在一般模式当中输入: r, R 为取代字符! (左下角出现 –Replace-)

- 如何由编辑模式跳回一般模式?

可以按下[Esc]

- 若上下左右键无法使用时，请问如何在一般模式移动光标?

[h, j, k, l]分别代表[左、下、上、右]

- 若 [pagedown] [pageup] 在一般模式无法使用时，如何往前或往后翻一页?

向下翻 [Ctrl] + [f]

向前翻 [Ctrl] + [b]

- 如何到本档案的最后一行、第一行;本行的第一个字符、最后一个字符?

分别为:G, 1G, 0, $

- 如何删除一行、n 行;如何删除一个字符?

分别为 dd, ndd, x 或 X (dG 及 d1G 分别表示删除到页首及页尾)

- 如何复制一行、n 行并加以贴上?

分别为 yy, nyy, p 或 P

- 如何搜寻 string 这个字符串?

?string (往前搜寻)

/string (往后搜寻)

- 如何取代 word1 成为 word2，而若需要使用者确认机制，又该如何?

:1,$s/word1/word2/g 或

:1,$s/word1/word2/gc (需要使用者确认)

- 如何读取一个档案 filename 进来目前这个档案?

:r filename

- 如何另存新档成为 newfilename?

:w newfilename

- 如何存档、离开、存档后离开、强制存档后离开?

:w; :q: :wq; :wq!

- 在 vi 底下作了很多的编辑动作之后，却想还原成原来的档案内容，应该怎么进行?

直接按下 :e! 即可恢复成档案的原始状态!

- 我在 vi 这个程序当中，不想离开 vi ，但是想执行 ls /home 这个指令，vi 有什么额外的功能可以达到这个目的:

事实上，可以使用[:! ls /home]不过，如果你学过后面的章节之后，你会发现，执行[ctrl + z]亦可暂时退出 vi 让你在指令列模式当中执行指令喔!

- 如何设定与取消行号？

:set nu

:set nonu

参考数据

- 关于 vim 是什么的『中文』说明：http://www.vim.org/6k/features.zh.txt。
- 李果正兄的：大家来学 vim (http://info.sayya.org/~edt1023/vim/)

认识 BASH Shell

最近更新日期：2005/08/30

文字模式 (command line) 这种指令下达的方式，在 Linux 里面，其实就相当于是 bash 的工具与接口！ 因为 Linux 就是以 bash 为预设的 shell 的!那么前几章我们都已经很快乐的进行了很多的指令下达啰～ 所以说， bash shell 根本就不难吧～是啦！只要能够熟悉的话，那么确实他也不是这么不可亲近的一项工具啊～ 这个章节中，鸟哥会由变量谈起，先讲到环境变量的功能与修改的问题， 然后会继续提到历史指令的运用。接下来，就会谈一下『数据流重导向』这个重要概念， 最后就是管线命令的利用啦！好好清一清脑门，准备用功去啰～ ^_^ 这个章节几乎是所有 command line 与未来主机维护与管理的重要基础，一定要好好仔细的阅读喔！

Bash shell

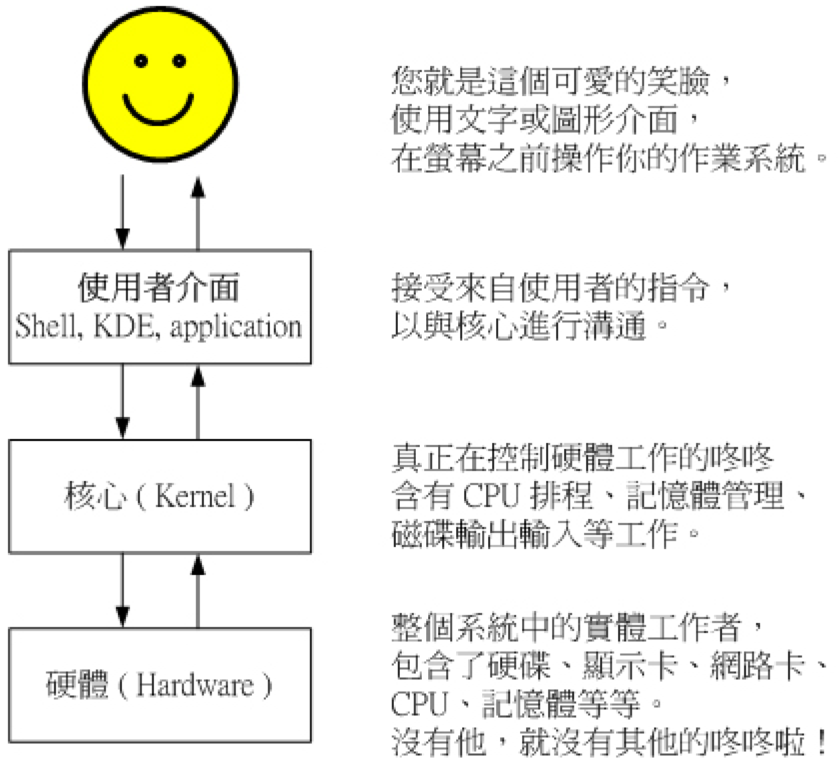
我们在前面的 什么是 Linux 那个章节当中，提到了， 管理整个硬件的其实是核心 (kernel)，那我们一般使用者 (user) 则是以 shell 来跟核心沟通～ 让核心达到我们所想要达到的工作目的。那么系统有多少 shell 可用呢？ 为什么我们要使用 bash 啊？！底下分别来谈一谈喔！

什么是 Shell?

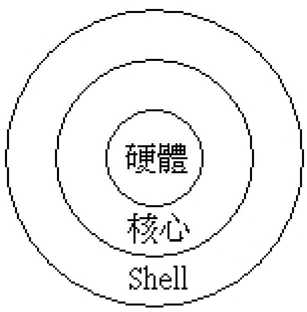
这应该是个蛮有趣的话题：『什么是 Shell ？』相信只要摸过计算机，对于操作系统（ 不论是 Linux 、Unix 或者是 Windows ）有点概念的朋友们大多听过这个名词，因为只要有『操作系统』那么就离不开 Shell 这个东西。不过，在讨论 Shell 之前，我们先来了解一下计算机的运作状况吧！ 举个例子来说：当你要计算机传输出来『音乐』的时候，你的计算机需要什么东西呢？

1. 当然就是需要你的硬件有『声卡芯片』这个硬件配备，否则怎么会有声音；
2. 操作系统的核心可以支持这个芯片组，当然还需要提供芯片的驱动程序啰；
3. 需要使用者（就是你）输入发生声音的指令啰！

这就是基本的一个输出声音的需要的步骤！那么也就是说，你必须要『输入』一个指令之后， 『硬件』才会透过你下达的指令来工作！嘿嘿！那么硬件如何知道你下达的指令呢？那就是 kernel （核心）的控制工作了！了解了吗？没错！也就是说，我们必须要透过『 Shell 』将我们输入的指令与 Kernel 沟通，好让 Kernel 可以控制硬件来正确无误的工作！ 基本上，我们可以透过底下这两张图来说明一下：



图一、硬件、核心与使用者的相关性图示



图二、硬件、核心与使用者的相关性图示

基本上，替我们工作的是『硬件』，而控制硬件的是『核心』，再来，我们使用者乃是利用『Shell』控制一些 kernel 提供的 『工具 (Utility)』来操控硬件替我们正确的工作。再进一步来说，由于 kernel 听不懂人类的语言，而人类也没有办法直接记得 kernel 的语言，所以两者的沟通就得藉由 shell 来支持了！(其实早期的 DOS 的文字接口也是使用 shell 来沟通呀！那个 shell 的名称就叫做 command.com ，还记得吗？ ^_^)

以字面上的意思来说， kernel 是『核心』的意思，而 Shell 是『壳』的意思，呵呵！也就是说， shell 是最外头的咚咚！而 kernel 乃是最内层的的咚咚啦！核心是操作系统的最底层的东西！ 这个核心里头包括了各种的支持硬件的工具！当然啰，如果你的硬件太新，而你的 kernel 并没有支持的话，那么很抱歉，你的 Shell 能力再怎么强，也没有办法使硬件工作的！ 这样可以了解了吗？呵呵！没错！使计算机主机工作的正是核心的任务，但是操作核心来替使用者工作的，却是 shell 喔！因此，有时候你的 shell 搞了老半天，硬件却不能工作的时候，请注意， 您的『核心』是否正确呢？阿！扯远了！这是 kernel 章节才要说的东西。

- 我干嘛要学习文字模式的 Shell 呢？

我们常常提到的 shell 其实是比较狭隘的定义，一般来说，在 Linux 里头，所谓的 shell 就是指 BASH 这个文字模式的 shell 啰。但是，广义的 shell 也可以是 KDE 之类的图形接口控制软件呢！ 因为他也可以帮我们与 kernel 进行沟通啊！不过，在鸟哥的 Linux 私房菜里面， 如果没有特别说明的话，那么我们的 shell 指的是比较狭义的，也就是文字模式的 shell 喔！

另外，鸟哥常常听到这个问题：『我干嘛要学习 shell 呢？ 不是已经有很多的工具可以提供我设定我的主机了？我为何要花这么多时间去学指令呢？不是以 X Window 按一按几个按钮就可以搞定了吗？为什么要这么麻烦？』唉～还是得一再地强调， X Window 还有 Web 接口的设定工具例如 webmin 是真的好用的家伙， 他真的可以帮助我们很简易的设定好我们的主机，甚至是一些很进阶的设定都可以帮我们搞定。

但是鸟哥在序章里面也已经提到过相当多次了， X Window 的接口虽然亲善，功能虽然强大，而 web 接口的工具也可以提供我们很友善的服务，但是毕竟他是将所有利用到的套件都整合在一起的一个套件而已，并非是一个完整的套件，所以某些时候当你升级或者是使用其它套件管理模块(例如 tarball 而非 rpm 档案等等)时，就会造成设定的困扰了。

此外，远程联机时，文字接口的传输速度一定比较快， 而且，较不容易出现断线或者是信息外流的问题，因此， shell 真的是得学习的一项工具。而且，他可以让您更深入 Linux ，更了解他， 而不是只会按一按鼠标而已！所谓『天助自助者！』多摸一点文字模式的东西，会让你与 Linux 更亲近呢！

有些朋友也很可爱，常会说：『我学这么多干什么？ 又不常用，也用不到！』嘿嘿！有没有听过『书到用时方恨少？』 当你的主机一切安然无恙的时候，您当然会觉得好像学这么多的东西一点帮助也没有呀！ 万一，某一天真的不幸给他中标了，您该如何是好？是直接重新安装？ 还是先追踪入侵来源后进行漏洞的修补？或者是干脆就关站好了？这当然涉及很多的考虑， 但就以鸟哥的观点来看，多学一点总是好的，尤其我们可以有备而无患嘛！甚至学的不精也没有关系，了解概念也就 OK 啦！毕竟没有人要您一定要背这么多的内容啦！了解概念就很了不起了！

此外，如果您真的有心想要将您的主机管理的好，那么良好的 shell 程序编写是一定需要的啦！就鸟哥自己来说，我管理的主机虽然还不算多， 只有区区不到十部，但是如果每部主机都要花上几十分钟来查阅他的 log file 以及相关的信息，那么我可能会疯掉！基本上，也太没有效率了！这个时候，如果能够藉由 shell 提供的命令重导向(或称数据流重导向)，以及管线命令，呵呵！那么我分析 log file 只要花费不到十分钟就可以看完所有的主机之重要信息了！相当的好用呢！

由于学习 shell 的好处真的是多多啦！所以，如果您是个系统管理员，或者有心想要管理系统的话，那么 shell 这个东西与 shell scripts 这个东西，真的真的有必要看一看！

系统的 shell 与 /etc/shells 功能

知道什么是 Shell 之后，那么我们来了解一下 Linux 使用的是哪一个 shell 呢？什么！哪一个？难道说 shell 不就是『一个 shell 吗？』哈哈！那可不！由于早年的 Unix 年代，发展者众，所以由于 shell 依据发展者的不同就有许多的版本，例如常听到的 Bourne SHell (sh) 、在 Sun 里头预设的 C SHell、 商业上常用的 K SHell、, 还有 TCSH 等等，每一种 Shell 都各有其特点。至于 Linux 使用的这一种版本就称为『 Bourne Again SHell (简称 bash) 』，这个 Shell 是 Bourne Shell 的增强版本，也是基准于 GNU 的架构下发展出来的呦！

在介绍 shell 的优点之前，先来说一说 shell 的简单历史吧：第一个流行的 shell 是由 Steven Bourne 发展出来的，为了纪念他所以就称为 Bourne shell ，或直接简称为 sh ！而后来另一个广为流传的 shell 是由柏克莱大学的 Bill Joy 设计依附于 BSD 版的 Unix 系统中的 shell ，这个 shell 的语法有点类似 C 语言，所以才得名为 C shell ，简称为 csh ！由于在学术界 Sun 主机势力相当的庞大，而 Sun 主要是 BSD 的分支之一，所以 C shell 也是另一个很重要而且流传很广的 shell 之一 (因为太多的程序设计师使用的就是 C 语言啦！)！(还记得我们在 Linux 是什么那一章提到的吧？ Sun 公司的创始人就是 Bill Joy，而 BSD 最早就是 Bill Joy 发展出来的啊！)。

那么目前我们的 Linux (以 FC4 为例) 有多少我们可以使用的 shells 呢？ 你可以检查一下 /etc/shells 这个档案，至少就有底下这几个可以用的 shells：

- /bin/sh (已经被 /bin/bash 所取代)
- /bin/bash (就是 Linux 预设的 shell)
- /bin/ksh (Kornshell 由 AT&T Bell lab. 发展出来的，兼容于 bash)
- /bin/tcsh (整合 C Shell ，提供更多的功能)
- /bin/csh (已经被 /bin/tcsh 所取代)
- /bin/zsh (基于 ksh 发展出来的，功能更强大的 shell)

由上面的说明中，我们大概可以发现，其实各主要 shell 的功能都差不多， 有的只是语法上面的不同而已。目前一般的使用者使用习惯上，似乎是以 bash 及 csh 为主要的两个 shell 。OK！这么多的 shell 我要使用哪一个啊？呵呵！使用 Linux 支持最广泛的 bash 就好了！ 不要想太多！另外，咦！为什么我们系统上的 shell 要写入 /etc/shells 这个档案啊？ 这是因为系统某些服务在运行过程中， 会去检查使用者能够使用的 shells ，而这些 shell 的查询就是藉由 /etc/shells 这个档案啰！

举例来说，某些 FTP 网站会去检查使用者的可用 shell ，而如果你不想要让这些使用者使用 FTP 以外的主机资源时，可能会给予该使用者一些怪怪的 shell，让使用者无法以其它服务登入主机。 这个时候，你就得将那些怪怪的 shell 写到 /etc/shells 当中了。举例来说，我们的 FC4 的 /etc/shells 里头就有个 /sbin/nologin 档案的存在，这个就是我们说的怪怪的 shell 啰～

那么，再想一想，我这个使用者什么时候可以取得 shell 来工作呢？还有， 我这个使用者预设会取得哪一个 shell 啊？！还记得我们在 首次进入 Linux -- 以文字方式登入 那个章节当中提到的登入动作吧？ 当我登入的时候，系统就会给我一个 shell 让我来工作了。 而这个登入取得的 shell 就记录在 /etc/passwd 这个档案内！这个档案的内容是啥？

```
[root@linux ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
.....(中间省略).....
```

如上所示，在每一行的最后一个数据，就是您登入后，可以取得的预设的 shell 啦！ 那你也会看到， root 是 /bin/bash ，不过，系统账号 bin 与 daemon 等等，就使用那个怪怪的 /sbin/nologin 啰～关于使用者这部分的内容，我们留在 账号管理 时提供更多的说明。

Bash shell 的功能

既然 /bin/bash 是 Linux 预设的 shell ，那么总是得了解一下这个玩意儿吧！ BASH 是怎么一回事呢？这个 shell 是 GNU 计划中重要的工具软件之一，目前也是 GNU 操作系统中标准的 shell ，他主要兼容于 sh ，并且依据一些使用者需求，而加强的 shell 版本，可以说目前几乎所有的 Linux distribution 都是使用 bash 作为管理核心的主要 shell ！因此，不论您使用的是那个 distribution ，你都难逃需要学习 bash 的宿命啦！那么这个 shell 有什么好处，干嘛 Linux 要使用他作为预设的 shell 呢？ BASH 主要的优点有底下几个：

- 命令编修能力（类似 DOS 的 doskey 功能）：

使用 bash 里头，个人认为相当棒的一个功能就是『他能记忆使用过的指令！』 这功能真的相当的棒！因为我只要在指令列按『上下键』就可以找到前一个输入的指令！而在很多 distribution 里头，预设的指令记忆功能可以到达 1000 个！也就是说， 你曾经下达过的指令都被记录下来了，记录的档案在你的家目录内的 .bash_history ！不过，需要留意的是， ~/.bash_history 记录的是前一次登入以前所执行过的指令， 而至于这一次登入所执行的指令都被暂存在暂内存中，当您成功的注销系统后，该指令记忆才会记录到 .bash_history 当中！

这有什么功能呢？最大的好处就是可以『查询曾经做过的举动！』， 如此可以知道你的执行步骤，那么就可以追踪您曾下达的指令，以作为除错的工具！ 但如此一来也有个烦恼，就是如果被黑客入侵了，那么他只要翻你曾经执行过的指令， 刚好你的指令又跟系统有关（例如直接输入 MySQL 的密码在指令列上面） 那么很容易就被破解你的 Linux 主机！所以，最好是将记录的指令数目减小一点较好！

- 命令与档案补全功能：

还记得我们在 首次进入 Linux 的热门按键 一节当中提到的 [tab] 这个按键吗？！这个按键的功能就是在 bash 里头才有的啦！常常在 bash 环境中使用 [tab] 是个很棒的习惯喔！因为至少可以让你 1)少打很多字； 2)确定输入的数据是正确的！ 使用 [tab] 按键的时机依据 [tab] 接在指令后或参数后而有所不同。我们再复习一次：

- [Tab] 接在一串指令的第一个字的后面，则为命令补全；
- [Tab] 接在一串指令的第二个字以后时，则为『档案补齐』！

所以说，如果我想要知道我的环境中，所有可以执行的指令有几个？ 就直接在 bash 的提示字符后面输入两个 [tab][tab] 就能够输出所有的可执行指令了。 那如果想要知道系统当中所有以 c 为开头的指令呢？就按下 c[tab][tab] 就好啦！ ^_^

是的！真的是很方便的功能，所以，有事没事，在 bash shell 底下，多按几次 [tab] 是一个不错的习惯啦！

- 命令别名(alias)设定功能：

假如我需要知道这个目录底下的所有档案（包含隐藏档）及所有的档案属性，那么我就必须要下达 ls -al 这样的指令列，唉！真麻烦，有没有更快的取代方式？呵呵！就使用命令别名呀！例如我最喜欢直接以 lm 这个自订的命令来取代上面的命令，也就是说， lm 会等于 ls -al 这样的一个功能，嘿！那么要如何作呢？就使用 alias 即可！你可以在指令列输入 alias 就可以知道目前的命令别名有哪些了！也可以直接下达命令来设定别名呦：

```
alias lm='ls -al'
```

- 工作控制(jobs)、前景背景控制：

这部分我们在之后的资源管理章节中会再提及！ 使用前、背景的控制可以让工作进行的更为顺利！至于工作控制(jobs)的用途则更广， 可以让我们随时将工作丢到背景中执行!而不怕不小心使用了 [Ctrl]+c 来停掉该程序！真是好样的！此外，也可以在单一登入的环境中，达到多任务的目的呢！

- Shell scripts 的强大功能：

在 DOS 年代还记得将一堆指令写在一起的所谓的『批次档』吧？在 Linux 底下的 shell scripts 则发挥的更为强大的功能，可以将您日常生活当中常需要下达的连续指令写成一个档案， 该档案并且可以透过对谈交互式的方式来进行主机的侦测工作！也可以藉由 shell 提供的环境变量及相关指令来进行设计，哇！整个设计下来几乎就是一个小型的程序语言了！该 scripts 的功能真的是超乎我的想象之外！以前在 DOS 底下需要程序语言才能写的东西，在 Linux 底下使用简单的 shell scripts 就可以帮你达成了！真的厉害！！这部分我们在后续章节再来谈！

- 万用字符！

除了完整的字符串之外， bash 还支持许多的万用字符来帮助使用者查询与指令下达。 举例来说，想要知道 /usr/X11R6/bin 底下有多少以 xt 为开头的档案吗？使用： ls -l /usr/X11R6/bin/xt* 就能够知道啰～此外，还有其它可供利用的万用字符， 这些都能够加快使用者的操作呢！

Bash shell 的内建命令： type

我们在首次进入 Linux 章节当中，提到关于 Linux 的在线说明文件 部分，也就是 man page 的内容，那么 bash 有没有什么说明文件啊？开玩笑～ 这么棒的东西怎么可能没有说明文件！请您在 shell 的环境下，直接输入 man bash 瞧一瞧， 嘿嘿！不是盖的吧！让您看个几天几夜也无法看完的 bash 说明文件，可是很详尽的数据啊！ ^_^

不过，在这个 man bash 所出现的 man page 当中，不知道您是否有察觉到，咦！ 怎么这个说明文件里面有其它的档案说明啊？举例来说，那个 cd 指令的说明就在这个 man page 内？ 然后我直接输入 man cd 时，怎么出现的画面中，最上方竟然出现一堆指令的介绍？？这是怎么回事？ 为了方便 shell 的操作，其实 bash 已经『内建』了很多指令了，例如上面提到的 cd ， 还有例如 umask 等等的指令，都是内建在 bash 当中的呢！

那我怎么知道这个指令是来自于外部指令(指的是其它非 bash 套件所提供的指令) 或是内建在 bash 当中的呢？ 嘿嘿！利用 type 这个指令来观察即可！举例来说：

```
[root@linux ~]# type [-tpa] name
参数：
    ：不加任何参数时，则 type 会显示出那个 name 是外部指令还是 bash 内建的指令！
-t  ：当加入 -t 参数时，type 会将 name 以底下这些字眼显示出他的意义：
      file    ：表示为外部指令；
      alias   ：表示该指令为命令别名所设定的名称；
      builtin ：表示该指令为 bash 内建的指令功能；
-p  ：如果后面接的 name 为指令时，会显示完整文件名(外部指令)或显示为内建指令；
-a  ：会将由 PATH 变量定义的路径中，将所有含有 name 的指令都列出来，包含 alias
范例：
范例一：查询一下 ls 这个指令是否为 bash 内建？
```

```
[root@linux ~]# type ls
ls is aliased to `ls --color=tty'
# 没有加上任何参数，仅列出 ls 这个指令的最主要使用情况
[root@linux ~]# type -t ls
alias
# -t 参数则仅列出 ls 这个指令的最主要使用情况说明
[root@linux ~]# type -a ls
ls is aliased to `ls --color=tty'
ls is /bin/ls
# 利用所有方法找出来的 ls 相关信息都会被列出来！

范例二：那么 cd 呢？
[root@linux ~]# type cd
cd is a shell builtin
```

透过 type 这个指令的用途，我们可以知道每个指令是否为 bash 的内建指令。 此外，由于利用 type 搜寻后面的名称时，如果后面接的名称并不能以执行档的状态被找到， 那么该名称是不会被显示出来的。举例来说，您的 FC4 应该不会有 vbird 这个指令吧？！ 输入 type -p vbird 看一下，果然没有输出任何数据！而如果您输入的是 type -p touch 呢？ 则会出现 /bin/touch ！呵呵！所以，这个 type 也可以用来作为类似 which 指令的用途啦！找指令用的！

指令的下达

我们在 首次进入 Linux 一节当中，已经提到过在 shell 环境下的指令下达方式，不过，因为这个部分实在很重要，所以，我们还是再次的提醒一次！

```
[root@linux ~]# command [-options] parameter1 parameter2 ...
                  指令     选项      参数(1)    参数(2)
说明：
0. 一行指令中第一个输入的绝对是『指令(command)』或『可执行档案』
1. command 为指令的名称，例如变换路径的指令为 cd 等等；
2. 中刮号[]并不存在于实际的指令中，而加入参数设定时，通常为 - 号，例如 -h；
   有时候完整参数名称会输入 -- 符号，例如 --help；
3. parameter1 parameter2.. 为依附在 option 后面的参数，
   或者是 command 的参数；
4. command, -options, parameter1.. 这几个咚咚中间以空格来区分，
   不论空几格 shell 都视为一格；
5. 按下 [Enter] 按键后，该指令就立即执行。[Enter] 按键为 <CR> 字符，
   他代表着一行指令的开始启动。
6. 指令太长的时候，可以使用 \ 符号来跳脱 [Enter] 符号，
   使指令连续到下一行。注意！ \ 后就立刻接特殊字符。
7. 在 Linux 系统中，英文大小写字母是不一样的。举例来说， cd 与 CD 并不同。
范例：
```

```
范例一：列出 /root 底下的各文件名称
[root@linux ~]# ls -al /root
[root@linux ~]# ls      -al       /root
# 不论指令与参数中间空几格，都是可以接受的！

范例二：如果指令太长的话，如何使用两行来输出？
[root@linux ~]# cp /var/spool/mail/root /etc/crontab \
> /etc/fstab /root
# 上面这个指令，就是将三个档案复制到 /root 这个目录下而已。不过，因为指令太长，
# 于是鸟哥就利用 \[Enter] 来将 [Enter] 这个按键『跳脱！』开来，让
# [Enter] 按键不再具有上述说明的第 5 点功能！好让指令继续在下一行输入。
# 需要特别留意， [Enter] 按键是紧接着反斜线 (\) 的，两者中间没有其它字符。
# 因为 \ 仅跳脱『紧接着的下一个字符』而已！所以，万一我写成：
# \ [Enter] ，亦即 [Enter] 与反斜线中间有一个空格时，则 \ 跳脱的是『空格键』
# 而不是 [Enter] 按键！这个地方请在仔细的看一遍！很重要！
# 如果顺利跳脱 [Enter] 后，下一行最前面就会主动出现 > 的符号，
# 您可以继续输入指令啰！也就是说，那个 > 是系统自动出现的，你不需要输入。
```

总之，当我们顺利的在终端机 (tty) 上面登入后， Linux 就会依据 /etc/passwd 档案的设定给我们一个 shell ，预设就是 bash ，然后我们就可以依据上面的指令下达方式来操作 shell， 之后，我们就可以透过 man 这个在线查询来查询指令的使用方式与参数说明， 很不错吧！那么我们就赶紧更进一步来操作 bash 这个好玩的东西啰！

Shell 的变量功能

在继续研究 BASH 之前，我们得要先就 变量 这个东西来讨论一番。 为什么要讨论变数呢？又，变数是啥玩意儿啊？！先来谈一谈国中数学好了，您是否依稀记得， 我们国中时候学过所谓的『 y = ax + b 』这东西？其中， y 是变量， x 则是这个变量的内容啊！ 讲的更简单一点，我们可以『用一个简单的 "字眼" 来取代另一个比较复杂或者是容易变动的数据』。这有什么好处啊？最大的好处就是『方便！』。

如果以 Linux 主机的运作来说明好了，因为在主机里面有太多的数据需要进行存取了， 而这些数据都是一些服务所必须的，例如某个名为 dmtsai 的账号，他的 mail 的存取路径预设是在 /var/spool/mail/dmtsai 、家目录预设在 /home/dmtsai 等等。那如果换了另外一个账号呢？ 假设另一个账号名称为 vbird ，你猜他的邮件与家目录在哪？应该是在 /var/spool/mail/vbird 与 /home/vbird 对吧！ 那么我们主机的邮件服务是否要记录好几个不同的路径啊？会不会太麻烦？这当然很麻烦啰～ 所以为了简化整个运作流程，我们就可以透过某个变量功能，让这个变量可以依据不同的使用者而变更内容，如此一来，系统的邮件服务只要依据那个变量去取得所需要的数据即可，就不需要记录不同的路径啰。

举例来说，我们每个账号的邮件信箱预设是以 MAIL 这个变量来进行存取的， 当 dmtsai 这个使用者登入时，他便会取得 MAIL 这个变量，而这个变量的内容其实就是 /var/spool/mail/dmtsai，那如果 vbird 登入呢？他取得的 MAIL 这个变量的内容其实就是 /var/spool/mail/vbird 。 而我们使用信件读取指令 mail 来读取自己的邮件信箱时，嘿嘿，这支程序可以直接读取 MAIL 这个变量的内容， 就能够自动的分辨出属于自己的信箱信件啰！这样一来，设计程序的设计师就真的很方便的啦！

当然我们可以改变这些个变量，但是如果该变量是直接深植于套件当中， 那么当你修改了某些参数之后，嘿嘿！你的套件就必须要『由原始码直接更新再编译』 才行！这样似乎很麻烦，所以啰，变量真的是很方便的啦！

Tips:
举个简单的例子来说， sendmail 的 smtp 存放 mail 路径是经由 /etc/profile 里头的：

MAIL="/var/spool/mail/$USER"

来设定的，而当我修改了上面这一个咚咚，然后重新开机之后，嘿嘿嘿嘿！ 我的邮件就可以存放到不同的路径去了！而且不会有问题！可以顺利的『在 Linux 主机上面』收发。然而问题发生在 pop3 这个服务上面，由于 pop3 的预设路径是在 source code 里头，而且就正是 /var/spool/mail 这个路径，也就是说，不论我怎么修正我的『变量』， pop3 都不为所动！唉～真惨，所以就无法直接以 pop3 来收信了（例如 OutLook 就不能工作了）！会发生密码不接受的问题呢！



再来继续讲到其它的变量功能好了，我们前面已经提到过很多次，能不能执行某个指令， 与 PATH 这个变量也有很大的关系的。举例来说，我们在任何地方下达 ls 这个指令时，系统就是透过 PATH 这个变量里面的内容所记录的路径顺序来搜寻指令的呢！如果在搜寻完 PATH 变量内的路径还找不到 ls 这个指令时，就会在屏幕上显示『 command not found 』的错误讯息了。

这些还都只是系统预设的变量的目的，如果是个人的设定方面的应用呢：例如你要写一个大型的 script （批次文件）时，有些数据因为可能由于使用者习惯的不同而有差异，比如说路径好了，由于该路径在 script 被使用在相当多的地方，如果下次换了一部主机，都要修改 script 里面的所有路径，那么我一定会疯掉！ 这个时候如果使用变量，而将该变量的定义写在最前面，后面相关的路径名称都以变量来取代，嘿嘿！那么你只要修改一行就等于修改整篇 script 了！方便的很！所以，良好的程序设计师都会善用变量的定义！ （ 这个部分我们在后续的 shell script 再次提及的！）

如果说的学理一点，那么由于在 Linux System 下面，所有的执行续都是需要一个执行码， 而就如同上面提到的，你『真正以 shell 来跟 Linux 沟通，是在正确的登入 Linux 之后！』这个时候你就有一个 bash 的执行程序，也才可以真正的经由 bash 来跟系统沟通啰！而在进入 shell 之前，也正如同上面提到的，由于系统需要一些变量来提供他数据的存取（或者是一些环境的设定参数值， 例如是否要显示彩色等等的），所以就有一些所谓的『环境变量』 需要来读入系统中了！这些环境变量例如 PATH、HOME、MAIL、SHELL 等等，都是很重要的， 为了区别与自订变量的不同，环境变量通常以大写字符来表示呢！

好了，那么我们就简单的来对『什么是变量』作个简单的定义好了： 『变量就是以一组文字或符号等，来取代一些设定或者是一串保留的数据！』， 例如：我设定了『myname』就是『VBird』，所以当你读取 myname 这个变量的时候，系统自然就会知道！哈！那就是 VBird 啦！最简单的例子可以取 PATH 来说明！如果你对于『相对路径与绝对路径』还有点印象的话， 那么应该晓得『要下达正确的指令，应该需要指定路径与文件名』才行！例如你的 ls 指令应该需要以『/bin/ls』来下达指令才对，那么为何你在任意的路径下都可以执行 ls 呢？而不需要指定路径呢？这是因为系统已经预设了一些『搜寻路径(PATH)』了， 所以当你需要执行一些指令的时候，系统就会依照该 PATH 的设定来进行指令的搜寻！而这个 PATH 就是所谓的变量了！

那么如何『显示变量』呢？这就需要使用到 echo 这个指令啦！

变量的取用与设定：echo, 变量设定规则, unset

说的口沫横飞的，也不知道『变量』与『变量代表的内容』有啥关系？ 当然啦，那我们就将『变量』的『内容』拿出来给您瞧瞧就好了。利用 echo 这个指令来取用变量， 但是，变量在被取用时，前面必须要加上 $ 才行，举例来说，要知道 PATH 的内容，该如何是好？

```
[root@linux ~]# echo $variable
[root@linux ~]# echo $PATH
/bin:/sbin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:/usr/X11R6/bin
[root@linux ~]# echo ${PATH}
```

变量的取用就如同上面的范例，利用 ehco 就能够读出，只是需要在变量名称前面加上 $ ， 或者是以 ${variable} 的方式来取用都可以！当然啦，那个 echo 的功能可是很多的， 我们这里单纯是拿 echo 来读出变量的内容而已，更多的 echo 使用，请自行给他 man echo 吧！ ^_^

例题一：请在屏幕上面显示出您的环境变量 HOME 与 MAIL：
答：

    echo $HOME
    echo $MAIL

OK！现在我们知道了变量与变量内的之间的相关性了，好了，那么我要如何『设定』或者是『修改』 某个变量的内容啊？！很简单啦！用『等号(=)』连接变量与他的内容就好啦！举例来说： 我要将 myname 这个变量名称的内容设定为 VBird ，那么：

```
[root@linux ~]# echo $myname
   <==这里并没有任何数据～因为这个变量尚未被设定！是空的！
[root@linux ~]# myname=VBird
[root@linux ~]# echo $myname
VBird  <==出现了！因为这个变量已经被设定了！
```

瞧！如此一来，这个变量名称 myname 的内容就带有 VBird 这个数据啰～ 而由上面的例子当中，我们也可以知道： 当一个变量名称尚未被设定时，预设的内容是『空』的。 另外，变量在设定时，还是需要符合某些规定的，否则会设定失败喔！ 这些规则如下所示啊！

1. 变量与变量内容以等号『=』来连结；
2. 等号两边不能直接接空格符；
3. 变量名称只能是英文字母与数字，但是数字不能是开头字符；
4. 若有空格符可以使用双引号『 " 』或单引号『 ' 』来将变量内容结合起来，但须要特别留意， 双引号内的特殊字符可以保有变量特性，但是单引号内的特殊字符则仅为一般字符；
5. 必要时需要以跳脱字符『 \ 』来将特殊符号 (如 Enter, $, \, 空格符, ' 等) 变成一般符号；
6. 在一串指令中，还需要藉由其它的指令提供的信息，可以使用 quote 『 ` command` 』；(特别特别注意，那个 ` 是键盘上方的数字键 1 左边那个按键，而不是单引号！)
7. 若该变量为扩增变量内容时，则需以双引号及 $变量名称 如：『 "$PATH":/home』继续累加内容；
8. 若该变量需要在其它子程序执行，则需要以 export 来使变量变成环境变量， 如『export PATH』；

9. 通常大写字符为系统预设变量，自行设定变量可以使用小写字符，方便判断 (纯粹依照使用者兴趣与嗜好) ；
10. 取消变量的方法为：『unset 变量名称』。

底下我们举几个例子来让您试看看，就知道怎么设定好您的变量啰！

```
范例一：设定一变量 name ，且内容为 VBird 。
[root@linux ~]# 12name=VBird
-bash: 12name=VBird: command not found  <==屏幕会显示错误！因为不能以数字开头！
[root@linux ~]# name = VBird  <==还是错误！因为有空白！
[root@linux ~]# name=VBird    <==OK 的啦！

范例二：承上题，若变量内容为 VBird's name 呢？
[root@linux ~]# name=VBird's name
# 因为单引号可以将 Enter 这个特殊字符取消，所以，您可以继续在下一行输入内容～
# 不过，这与我们要达到的功能不同，所以，算是失败的啦！
[root@linux ~]# name="VBird's name"  <==OK 的啦！
[root@linux ~]# name=VBird\'s\ name
# 利用反斜线 (\) 跳脱特殊字符，例如单引号与空格键，这也是 OK 的啦！

范例三：我要在 PATH 这个变量当中『累加』:/home/dmtsai/bin 这个目录
[root@linux ~]# PATH=$PATH:/home/dmtsai/bin
[root@linux ~]# PATH="$PATH":/home/dmtsai/bin
# 上面这两种格式在 PATH 里头的设定都是 OK 的！但是底下的例子就不见得啰！

范例四：呈范例三，我要将 name 的内容多出 "yes" 呢？
[root@linux ~]# name=$nameyes
# 知道了吧？如果没有双引号，那么变量成了啥？name 的内容是 $nameyes 这个变量！
# 呵呵！我们可没有设定过 nameyes 这个变量呐！所以，应该是底下这样才对！
[root@linux ~]# name="$name"yes
[root@linux ~]# name=${name}yes

范例五：如何让我刚刚设定的 name=VBird 可以用在下个 shell 的程序？
[root@linux ~]# name=VBird
[root@linux ~]# bash        <==进入到所谓的子程序
[root@linux ~]# echo $name  <==嘿嘿！并没有刚刚设定的内容喔！
[root@linux ~]# exit        <==离开刚刚的子程序
[root@linux ~]# export name
[root@linux ~]# bash        <==进入到所谓的子程序
[root@linux ~]# echo $name  <==出现了设定值了！
[root@linux ~]# exit        <==离开刚刚的子程序
# 什么是『子程序』呢？就是说，在我目前这个 shell 的情况下，
# 去启用另一个新的 shell ，新的那个 shell 就是子程序啦！在一般的状态下，
# 父程序的自订变量是无法在子程序内使用的。但是透过 export 将变量变成
```

```
# 环境变量后，就能够在子程序底下应用了！很不赖吧！至于程序的相关概念，
# 我们会在『程序与资源管理』章节当中提到的喔！

范例六：如何进入到您目前核心的模块目录？
[root@linux ~]# cd /lib/modules/`uname -r`/kernel
# 每个操作系统核心版本都不相同，以 FC4 为例，他的预设核心版本是
# 2.6.11-1.1369_FC4 所以，他的模块目录在 /lib/modules/2.6.11-1.1369_FC4/kernel 。
# 因为每个 distributions 的这个值都不相同，但是我们却可以利用 uname -r 这个指令
# 先取得版本信息，所以啰，就可以透过上面指令当中的内含指令 `uname -r`
# 先取得版本输出到 cd .. 那个指令当中，就能够顺利的进入目前核心的驱动程序所放置
# 的目录啰！很方便吧！

范例七：取消刚刚设定的 name 这个变量内容
[root@linux ~]# unset name
```

根据上面的案例你可以试试看！就可以了解变量的设定啰！这个是很重要的呦！请勤加练习！！ 其中，较为重要的一些特殊符号的使用啰！例如单引号、双引号、跳脱字符、钱字号、quote 符号等等，底下的例题想一想吧！

例题二：在变量的设定当中，单引号与双引号的用途有何不同？

答：

单引号与双引号的最大不同在于双引号仍然可以保有变量的内容，但单引号内仅能是一般字符 ，而不会有特殊符号。我们以底下的例子做说明：假设您定义了一个变量，name=VBird ，现在想以 name 这个变量的内容定义出 myname 显示 VBird its me 这个内容，要如何订定呢？

```
[root@linux ~]# name=VBird
[root@linux ~]# echo $name
VBird
[root@linux ~]# myname="$name its me"
[root@linux ~]# echo $myname
VBird its me
[root@linux ~]# myname='$name its me'
[root@linux ~]# echo $myname
$name its me
```

发现了吗？没错！使用了单引号的时候，那么 $name 将失去原有的变量内容， 仅为一般字符的显示型态而已！这里必需要特别小心在意！

例题三：在指令下达的过程中， quote （ ` ） 这个符号代表的意义为何？

答：

在一串指令中，在 ` 之内的指令将会被先执行，而其执行出来的结果将做为外部的输入信息！例如 uname -r 会显示出目前的核心版本，而我们的核心版本在 /lib/modules

里面，因此，你可以先执行 uname -r 找出核心版本，然后再以『 cd 目录』到该目录下，当然也可以执行如同上面范例六的执行内容啰。

另外再举个例子，我们也知道， locate 指令可以列出所有的相关档案档名，但是， 如果我想要知道各个档案的权限呢？举例来说，我想要知道每个 crontab 相关档名的权限：

```
[root@linux ~]# ls -l `locate crontab`
```

如此一来，先以 locate 将文件名数据都列出来，再以 ls 指令来处理的意思啦！瞭了吗？ ^_^

变数的用途

我们知道 PATH 这个变量是我们在执行指令的时候，所需要具备的指令搜寻目录数据， 没有他，我们就得要使用绝对路径来下达指令才行。当然，还有很多变量都有他特别的意义存在。 除此之外，『我为何需要设定变量』呢？ 要跟大家介绍这个『变量』，当然是因为他有相当程度的意义存在的啊！ 底下就跟大家介绍一下，鸟哥设定变量的时机喔！

> 我的案例一：最简单的例子就是 『简化路径名称』啰！以鸟哥为例，我的工作在 Unix 系统之下进行一些数值模式的仿真工作，偏偏由于数据量太大， 为了怕日后忘记这个目录的内容与主要的意义，所以我的档名都取的很长， 偏偏在执行模式的过程中，常常会切换目录！我哩ㄌㄟ，光是打那几行路径名称就快要疯掉了！ 所以我就设定那几行目录名称成为一个四个字符的变量，如此一来我只要输入『 cd $VARI 』这个指令，嘿嘿！马上就移动到该路径下了！很方便吧！当然变量的意义还不止于此， 不过这是最简单的实例说明啰！
> 我的案例二：另外一个常常需要变量的咚咚是在 scripts 里面，例如我写的一个侦测登录文件的小程序 logfile.sh 这个咚咚， 由于里头常常需要用到『储存路径』，偏偏可能每个人的存取路径都不太一样， 而如果要修改存取路径的话，嘿嘿！好几十行要同时修改呢！还可能会改错！ 那么我只要定义一个变量，然后后续的所有数据都使用这个变量的内容！嘿嘿！ 那么只要大家修改了这个变量的内容（只要一行），后续的动作就不需要修正了！这个动作常在程序或者是 script 当中看到的！

所以啰，有很多的时候为了方便或者是使用于 scripts 的意义，我们必须要设定变量！ 当然啰，如果是跟系统终端机环境有关的设定值，很多也是利用变量来帮助达成的～ 底下我们就来谈一谈所谓的『环境变量』吧！

环境变量的功能

环境变量可以帮我们达到很多功能～包括家目录的变换啊、提示字符的显示啊、执行文件搜寻的路径啊等等的， 还有很多很多啦！那么，既然环境变量有那么多的功能，问一下，目前我的 shell 环境中， 有多少变量啊？！呵呵！我们可以利用两个指令来查阅，分别是 env 与 export 呢！

- 一些环境变量的说明： env

```
范例一：列出目前的 shell 环境下的所有环境变量与其内容。
[root@linux ~]# env
HOSTNAME=linux.dmtsai.tw    <== 这部主机的主机名称
SHELL=/bin/bash             <== 目前这个环境下，使用的 Shell 是哪一个程序？
TERM=xterm                  <== 这个终端机使用的环境是什么类型
HISTSIZE=1000               <== 这个就是『记录指令的笔数』在 FC4 预设可记录 1000 笔
USER=root                   <== 使用者的名称啊！
LS_COLORS=no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;01:cd=40;33;01:
or=01;05;37;41:mi=01;05;37;41:ex=00;32:*.cmd=00;32:*.exe=00;32:*.com=00;32:*.btm=0
0;32:*.bat=00;32:*.sh=00;32:*.csh=00;32:*.tar=00;31:*.tgz=00;31:*.arj=00;31:*.taz=
00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:*.gz=00;31:*.bz2=00;31:*.bz=00;3
1:*.tz=00;31:*.rpm=00;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp=00;35:*.xbm=00
;35:*.xpm=00;35:*.png=00;35:*.tif=00;35: <== 一些颜色显示
ENV=/root/.bashrc           <== 使用的个人环境设定档
MAIL=/var/spool/mail/root   <== 这个使用者所取用的 mailbox 位置
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:/usr/local/sbin:
/root/bin                   <== 不再多讲啊！是执行文件指令搜寻路径
INPUTRC=/etc/inputrc        <== 与键盘按键功能有关。可以设定特殊按键！
PWD=/root                   <== 目前使用者所在的工作目录 (利用 pwd 取出！)
LANG=en_US.UTF-8            <== 这个与语系有关，底下会再介绍！
HOME=/root                  <== 这个使用者的家目录啊！
_=/bin/env                  <== 上一次使用的指令的最后一个参数(或指令本身)
```

env 是 environment (环境) 的简写啊～ 上面的例子当中，是列出来所有的环境变量。当然，如果使用 export 也会是一样的内容～ 只不过， export 还有其它额外的功能就是了，我们等一下再提这个 export 指令。 那么上面这些变量有些什么功用呢？底下我们就一个一个来分析分析！

- HOME ： 代表使用者的家目录。还记得我们可以使用 cd ~ 去到使用者的家目录吗？或者利用 cd 就可以直接回到使用者家目录了。那就是取用这个功能啦～ 有很多程序都可能会取用到这个变量的值喔！
- SHELL ： 告知我们，目前这个环境使用的 SHELL 是哪支程序？ 如果是 bash 的话，预设是 /bin/bash 的啦！
- HISTSIZE ： 这个与『历史命令』有关，亦即是， 我们曾经下达过的指令可以被系统记录下来，而记录的『笔数』则是由这个值来设定的。
- ENV ： 这个使用者所使用的个人化环境设定档的读取档案。
- MAIL ： 当我们使用 mail 这个指令在收信时，系统会去读取的邮件信箱档案 (mailbox)。
- PATH ： 就是执行文件搜寻的路径啦～目录与目录中间以冒号(:)分隔， 由于档案的搜寻是依序由 PATH 的变量内的目录来查询，所以，目录的顺序也是重要的喔。
- LANG ： 这个重要！就是语系档案啰～很多数据都会用到他， 举例来说，当我们在启动某些 perl 的程序语言档案时，他会主动的去分析语系数据文件， 如果发现有他无法解析的编码语系，可能会产生错误喔！一般来说，我们中文编码通常是 zh_TW.Big5 或者是 zh_TW.UTF-8，这两个编码偏偏不容易被解译出来，所以，有的时候，可能需要修订一下语系数据。 这部分我们会在下个小节做介绍的！
- RANDOM ： 这个玩意儿就是『随机随机数』的变量啦！目前大多数的 distributions 都会有随机数产生器，那就是 /dev/random 这个档案。我们可以透过这个随机数档案相关的变量 ($RANDOM)

来随机取得随机数值喔。在 BASH 的环境下，这个 RANDOM 变量的内容，介于 0~32767 之间，所以，你只要 echo $RANDOM 时，系统就会主动的随机取出一个介于 0~32767 的数值。万一我想要使用 0~9 之间的数值呢？呵呵～利用 declare 宣告数值类型， 然后这样做就可以了：

```
[root@linux ~]# declare -i number=$RANDOM*10/32767 ; echo $number
8   <== 此时会随机取出 0~9 之间的数值喔！
```

大致上是有这些环境变量啦～里面有些比较重要的参数，在底下我们都会另外进行一些说明的～

- 其它所有的变量说明： set

而除了这些环境变量之外，还有没有什么重要的变量呢？当然有啊！ 我们在 bash 的环境下，其实还有一些挺重要的变量，这些变量是『在这个 shell 环境下有效』的， 如果是在『子程序』，这些变量值就不会相同了。 那么如何观察目前 shell 环境下的所有变量呢？很简单啊，就用 set 即可！set 这个指令除了会将环境变量列出来之外，其它我们的自订变量，与所有的变量，都会被列出来喔！信息多好多。 底下仅列出几个重要的内容。

```
[root@linux ~]# set
BASH=/bin/bash              <== bash 的主程序放置路径
BASH_VERSINFO=([0]="3" [1]="00" [2]="16" [3]="1" [4]="release"
[5]="i386-redhat-linux-gnu")        <== bash 的版本啊！
BASH_VERSION='3.00.16(1)-release' <== bash 的版本啊！
COLORS=/etc/DIR_COLORS.xterm        <== 使用的颜色纪录档案
COLUMNS=115                 <== 在目前的终端机环境下，使用的字段有几个字符长度
HISTFILE=/root/.bash_history        <== 历史命令记录的放置档案，隐藏档
HISTFILESIZE=1000           <== 存起来(与上个变量有关)的档案之指令的最大纪录笔数。
HISTSIZE=1000               <== 目前环境下，可记录的历史命令最大笔数。
HOSTTYPE=i386               <== 主机安装的软件主要类型。我们用的是 i386 兼容机器软件
IFS=$' \t\n'                <== 预设的分隔符
LINES=35                    <== 目前的终端机下的最大行数
MACHTYPE=i386-redhat-linux-gnu      <== 安装的机器类型
MAILCHECK=60                <== 与邮件有关。每 60 秒去扫瞄一次信箱有无新信！
OLDPWD=/home                <== 上个工作目录。我们可以用 cd - 来取用这个变量。
OSTYPE=linux-gnu            <== 操作系统的类型！
PPID=20046                  <== 父程序的 PID (会在后续章节才介绍)
PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}\007"'
                            <== 上面这个是命令提示字符！与底下也有关。
PS1='[\u@\h \W]\$ '         <== PS1 就厉害了。这个是命令提示字符，也就是我们常见的
                                [root@linux ~]# 或 [dmtsai ~]$ 的设定值啦！可以更动的！
RANDOM=13586                <== 随机数啊！上面已经提过啰～
SUPPORTED=zh_TW.UTF-8:zh_TW:zh:en_US.UTF-8 <== 本系统所支持的语系
name=VBird                  <== 刚刚设定的自订变量也可以被列出来喔！
$                           <== 目前这个 shell 所使用的 PID
?                           <== 刚刚执行完指令的回传值。
```

一般来说，不论是否为环境变量，只要跟我们目前这个 shell 的操作接口有关的变量， 通常都会被设定为大写字符，也就是说，『基本上，在 Linux 预设的情况中，使用{大写的字母}来设定的变量一般为系统

内定需要的变量』。

使用 set 除了会将系统的默认值秀出来之外，连带的所有的你自己设定的变量也会被秀出来！ 同时需要注意的是，若当时有相当多人同时在在线的话，那么 你的变量只能给自己使用 (除非改的是系统的预设参数档，如 /etc/profile)，而不会干扰到别人的！就如同前面所说的， 由于你登入 Linux 之后会取得一个 PID ，而你的设定将只对这个 PID 与子程序有关！此外， 这次登入所进行的变量设定，如果没有更动到设定档， 那么这次设定的变量在下次登入时将被取消掉 (因为程序 PID 不见啰！) ！所以啰， 如果你想要你的变量每次都能在你登入的时候自动就设定好了，那么就必须将你的设定写入登入时加载的设定档！ (更多的程序相关的说明，不要急～我们会在后面的 程序与资源管理 当中好好的提一提的！)

OK！OK！那么上头那些变量当中，有哪些是比较重要的？大概有这几个吧！

- PS1：(提示字符的设定)

  这是 PS1 (数字的 1 不是英文字母！)，这个东西就是我们的『命令提示字符』啊！ 当我们每次按下 [Enter] 按键去执行某个指令后，最后要再次出现提示字符时， 就会主动去读取这个变数值了。上头 PS1 内显示的是一些特殊符号，每个版本 bash 的 PSI 变量内的特殊符号可能有些许的差异， 你应该主动的以 man bash 去查询一下相关的变数。底下我列出 FC4 的环境下， 预设的 bash 的 PS1 变量内的特殊符号代表意义：

  - \d ：代表日期，格式为 Weekday Month Date，例如 "Mon Aug 1"
  - \H ：完整的主机名称。举例来说，鸟哥的练习机 linux.dmtsai.tw ，那么这个主机名称就是 linux.dmtsai.tw
  - \h ：仅取主机名称的第一个名字。以上述来讲，就是 linux 而已， .dmtsai.tw 被省略。
  - \t ：显示时间，为 24 小时格式，如： HH:MM:SS
  - \T ：显示时间，12 小时的时间格式！
  - \A ：显示时间，24 小时格式， HH:MM
  - \u ：目前使用者的账号名称；
  - \v ：BASH 的版本信息；
  - \w ：完整的工作目录名称。家目录会以 ~ 取代；
  - \W ：利用 basename 取得工作目录名称，所以仅会列出最后一个目录名。
  - \# ：下达的第几个指令。
  - \$ ：提示字符，如果是 root 时，提示字符为 # ，否则就是 $ 啰～

  OK！所以，由预设的 PS1 内容为： '\[\u@\h \W\]\$ ' 就可以了解为何我们的提示字符会是： [root@linux ~]# 了吧！好了，那么假设我想要有类似底下的提示字符：

  ```
  [root@linux /home/dmtsai 16:50 #12]#
  ```

，那个 # 代表第 12 次下达的指令。 那么应该如何设定 PS1 呢？可以这样啊：

```
[root@linux home]# PS1='[\u@\h \w \A #\#]\$ '
[root@linux /home 17:02 #85]#
# 看到了吗？提示字符变了！变的很有趣吧！其中，那个 #85 比较有趣，
# 如果您按下 [Enter] 后，该数字就会增加喔！为啥？上面有说明ㄇㄟ！
```

- $：(关于本 shell 的 PID)

  其实这个咚咚代表的是『目前这个 Shell 的执行绪代号』，亦即是所谓的 PID (Process ID)。 更多的程序观念，我们会在第四章的时候提及。想要知道我们的 shell 的 PID ，就可以： echo $$ 即可！
- ?：(关于上个执行指令的回传码)

  虾密？问号也是一个特殊的变数？没错！在 bash 里面这个变量可重要的很！ 这个变数是：『上个执行的指令所回传的值』， 上面这句话的重点是『上一个指令』与『回传值』两个地方。当我们执行某些指令时， 这些指令都会回传一个执行后的代码。一般来说，如果成功的执行该指令，则会回传一个 0 值，如果执行过程发生错误，就会回传『错误代码』才对！一般就是以非为 0 的数值来取代。 我们以底下的例子来看看：

```
[root@linux ~]# echo $SHELL
/bin/bash
[root@linux ~]# echo $?
0
# 因为上个指令执行过程中，并没有错误，为成功的执行完毕，所以回传 0 。
[root@linux ~]# 12name=VBird
-bash: 12name=VBird: command not found
[root@linux ~]# echo $?
127
# 发生错误啦！所以 echo $? 时，就会出现错误的代码！
# 我们可以利用这个代码来搜寻错误的原因喔！
[root@linux ~]# echo $?
0
# 咦！怎么又变成正确了？这是因为 "?" 只与『上一个执行指令』有关，
# 所以，我们上一个指令是执行『 echo $? 』，当然没有错误，所以是 0 没错！
```

- OSTYPE, HOSTTYPE, MACHTYPE：(主机硬件与核心的等级)

这几个东西与程序的安装有关。我们在『Linux 主机规划』 里面提到过关于主机的等级方面的问题，当我们在安装软件的时候， 需要透过编译器来将原始码编译成为二进制的档案 (binary file)。但是， 我们可以针对硬件的配备来进行编译的最佳化，此时，这些参数就可以被用到了！基本上，目前主要的 distribution 都是针对 i386 亦即最低等级的机器进行最佳化， 这样才能够安装在较高阶的机器上，如果以 686 的机型来最佳化， 那么，可就无法向下兼容的喔！(早期的 OpenLinux 是针对 686 机器来释出软件， 所以，当时的 OpenLinux 是无法安装在 P-166 的机器上的。)

自订变量转成环境变量： export

好了，上面我们环境变量也提过了，一些自订变量也提过了，那么，这两者有啥不同？ 他的不同处，我们在 变量设定规则 当中稍微提过， 主要是由于变量可否被子程序所引用。

当你取得一个 bash 之后，亦即得到了一个程序了，但是若你再次的执行一次 bash ，那么你将进入『子程序』，这个程序的概念我们在资源管理章节中再详谈，这里您先有个概念即可。 那么由于您已经进入了该子程序，所以在父程序中的自订变量设定将不再继续的存在。 会存在子程序中的，仅有『环境变量』。

换个角度来想，也就是说，如果我能将自订变量变成环境变量的话，那不就可以让该变量值继续存在于子程序了？ 呵呵！没错！此时，那个 export 指令就很有用啦！ 如您想要让该变量内容继续的在子程序中使用，那么就请执行：

　　export 变数

这个东西用在『引用他人的档案或者其它程序』时，相当的重要的！ 尤其像鸟哥常常两三个档案互相引用来引用去的，如果忘记设定 export 的话，那么不同的档案中的相同变量值，将需要一再地重复设定才行！所以，我只要在头一个档案使用 export 的话，那么后续的档案引用时，将会把该变量内容读进来！好用的很，如果仅下达 export 而没有接变量时，那么此时将会把所有的『环境变量』秀出来喔！例如：

```
[root@linux ~]# export
declare -x ENV="/root/.bashrc"
declare -x HISTSIZE="1000"
declare -x HOME="/root"
declare -x HOSTNAME="linux.dmtsai.tw"
declare -x INPUTRC="/etc/inputrc"
declare -x LANG="en_US.UTF-8"
declare -x MAIL="/var/spool/mail/root"
declare -x SHELL="/bin/bash"
# 很多都直接省略了！不然....重复性太高，浪费版面～ ^_^
```

语系档案的变量 (locale)

还记得我们在首次进入 Linux 那个章节里面提到的，关于语系编码的问题吗？ 就是当我们使用 man command 的方式去查询某个数据的说明文件时，该说明档的内容可能会因为我们使用的语系， 而产生一些乱码。另外，利用 ls 查询档案的时间时，也可能会有乱码出现在时间的部分。 那个问题其实就是语系的问题啦。

目前大多数的 Linux distributions 已经都是支持万国码，此外，也都支持大部分的语言语系了。 这有

赖于 i18n 支援的帮助呢！ 那么我们的 Linux 到底支持了多少的语系呢？这可以由 locale 这个指令来查询到喔！

```
[root@linux ~]# locale -a
aa_DJ
aa_DJ.iso88591
en_US
en_US.iso88591
en_US.iso885915
en_US.utf8
zh_TW
zh_TW.big5
zh_TW.euctw
zh_TW.utf8
# 其实输出的内容有很多，鸟哥将一些信息舍弃了～
# 从上面的输出中，我们也不难看出，系统是有支持 big5, utf8 等中文语系数据的！
```

中文语系至少支持了两种以上的编码，一种是目前还是很常见的 big5 ，另一种则是越来越热门的 utf-8 编码。 那么我们如何修订这些编码呢？其实可以透过底下这些变量的说：

```
[root@linux ~]# LANG          <==主语言的环境
[root@linux ~]# LC_CTYPE      <==字符辨识的编码
[root@linux ~]# LC_NUMERIC    <==数字系统的显示讯息
[root@linux ~]# LC_TIME       <==时间系统的显示数据
[root@linux ~]# LC_COLLATE    <==字符串的比较与排序等
[root@linux ~]# LC_MONETARY   <==币值格式的显示等
[root@linux ~]# LC_MESSAGES   <==讯息显示的内容，如菜单、错误讯息等
[root@linux ~]# LC_ALL        <==语言环境的整体设定。
```

基本上，你可以逐一设定每个与语系有关的变量数据，但事实上，如果其它的语系变量都未设定， 且您有设定 LANG 或者是 LC_ALL 时，则其它的语系变量就会被这两个变量所取代！这也是为什么我们在 FC4 当中，通常仅设定 LANG 这个变量而已！因为他是最主要的设定变量。 好了，那么你应该要觉得奇怪的是，为什么在 Linux 主机的终端机接口 (tty1 ~ tty6) 的环境下，如果 LANG=zh_TW.big5 这个设定值生效后，使用 man 或者其它讯息输出时， 都会有一堆乱码，尤其是使用 ls -l 这个参数时？

因为在 Linux 主机的终端机接口下，那个环境是无法显示像中文这么复杂的编码的文字， 所以，就会产生乱码了。也就是如此，所以，我们才会必须要在 tty1 ~ tty6 的环境下， 加装一些中文化接口的软件，才能够看到中文啊！不过，如果您是在 Windows 主机以远程联机服务器的软件联机到主机的话，那么，嘿嘿！其实文字接口确实是可以看到中文的。 所以，此时反而您得要在 LANG 设定中文编码才好呢！

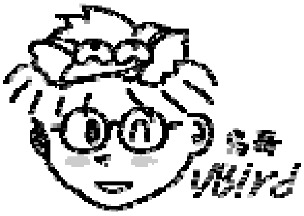
无论如何，如果发生一些乱码的问题，那么设定系统里面保有的语系编码， 例如： en_US 或 en_US.utf8 等等的设定，应该就 OK 的啦！好了，那么系统预设支持多少种语系呢？ 当我们使用 locale 时，系统是列出目前 Linux 主机内保有的语系档案， 这些语系档案都放置在： /usr/lib/locale/ 这个目录中。 但是，目前的这个 shell 环境所支持的语系，则是要看 SUPPORTED 这个变数才对喔！

那么，如果我想要修订系统的语系支持呢？可以修订 /etc/sysconfig/i18n 这个档案呢！ 这个档案的内容有点像这样：

```
[root@linux ~]# vi /etc/sysconfig/i18n
LANG="en_US.UTF-8"
SYSFONT="latarcyrheb-sun16"
SUPPORTED="zh_TW.UTF-8:zh_TW:zh:en_US.UTF-8"
```

你可以在这个档案当中加入 LC_TIME 或者其它语系相关变量的设定内容， 也可以直接修改 LANG 那个变量即可啊！ ^_^ 但，事实上，我们还可以透过个人的环境设定档来设定 LANG 呢！ 如此一来，则不必修订系统的语系档案，比较安全啦！

Tips:
假设你用 vi 编辑一个纯文字文件，这个纯文字文件在编辑的时候，是在 Windows 上面编辑的， 那么这个档案的预设编码应该是以 zh_TW.big5 所编辑的才对。但是，如果你的 shell 环境中，却是使用 LANG=en_US 时，则当你编辑该档案时，就可能会看到『乱码』， 或者输入的中文可能会变成『乱码』了。此时，只要你离开 vi ，然后执行 LANG=zh_TW.big5 ， 然后再重新以 vi 编辑该档案，呵呵！应该就能够看到中文啦！但是请注意， 这个方法当然不适用 tty1 ~ tty6 的环境，原因上面已经提过啰～ 仅适合以类似 putty 软件由 Windows 计算机联机到 linux 主机上的做业！



变量的有效范围

虾密？？变量也有使用的『范围』？没错啊～我们在上头的 export 指令说明中，就提到了这个概念了。如果在跑程序的时候，有父程序与子程序的不同程序关系时， 则『变量』可否被引用是 export 有关。被 export 后的变量，我们可以称他为『环境变量』！ 环境变量可以被子程序所引用，但是其它的自订变量内容就不会存在于子程序中。也就是说： 我们自行设定的变量，只在目前这个 shell 环境当中存在， 在子程序中将不会存在此一变量。除非使用 export 将自订变量变成环境变量。

其实除了 shell 的父、子程序外，在脚本(scripts)的编写当中，由于有的软件会使用到 2 个以上的 scripts 做为一个完整的套件！也就是说，假如你有两支程序，一支为 scripts1.sh 以及 scripts2.sh ，而 scripts2.sh 会去引用 scripts1.sh 的变数，这个时候，嘿嘿！你在 scripts1.sh 当中设定的变量请『千万记得以 export 设定』， 否则你的变量将无法在两个 scripts 之间互相被引用喔！当这个 scripts 执行完毕之后，刚刚在 scripts 当中设定的变量也就『失效了！』。

其实，要了解不同程序之间变量的变换，应该要先了解『程序』的概念比较好， 但是我们还没有讲到.....没关系～等你念到程序章节后，还可以再回来好好的看一看。 基本上，环境变量可以让子程序继续引用的原因，是因为：

- 当启动一个 shell ，操作系统分配一记忆区块给 shell 使用，此区域之变量可以让子程序存取；
- 利用 export 功能，可以让变量的内容写到上述的记忆区块当中(环境变量)；
- 当加载另一个 shell 时 (亦即启动子程序，而离开原本的父程序了)，子 shell 可以将父 shell 的环境变量所在的记忆区块导入自己的环境变量区块当中。

透过这样的关系，我们就可以让某些变量可以在相关的程序之间存在，以帮助自己更方便的操作环境喔！

变量键盘读取、数组与宣告： read, array, declare

我们上面提到的变量设定功能，都是直接由指令列直接设定的，那么，可不可以让使用者能够经由键盘输入？ 什么意思呢？是否记得某些程序执行的过程当中，会等待使用者输入 "yes/no" 之类的讯息啊!? 在 bash 里面也有相对应的功能喔！此外，我们还可以宣告这个变量的属性， 例如：数组或者是数字等等的。底下就来看看吧！

- read

要读取来自键盘输入的变量，就是用 read 这个指令了。这个指令最常被用在 shell script 的撰写当中，以跟使用者进行对谈。关于 script 的写法，我们会在后面章节介绍，底下先来瞧一瞧 read 的相关语法吧！

```
[root@linux ~]# read [-pt] variable
参数：
-p  ：后面可以接提示字符！
-t  ：后面可以接等待的『秒数！』这个比较有趣～不会一直等待使用者啦！
范例：

范例一：让使用者由键盘输入一内容，将该内容变成 atest 变量
[root@linux ~]# read atest
This is a test
[root@linux ~]# echo $atest
This is a test

范例二：提示使用者 30 秒内输入自己的大名，将该输入字符串做成 named 变量
[root@linux ~]# read -p "Please keyin your name: " -t 30 named
Please keyin your name: VBird Tsai
[root@linux ~]# echo $named
VBird Tsai
```

read 之后不加任何参数，直接加上变量名称，那么底下就会主动出现一个空白行，等待您输入。 如果加上 -t 后面接秒数之后，例如上面的范例当中，那么 30 秒之内没有任何动作时，该指令就会自动略过了～如果是加上 -p ，嘿嘿！后面就会有比较多可以用的提示字符给我们参考！ 在指令的下达里面，比较美观啦！ ^_^

- declare / typeset

declare 或 typeset 是一样的功能，就是在宣告变量的属性。如果使用 declare 后面并没有接任何参数，那么 bash 就会主动的将所有的变量名称与内容通通叫出来，就好像使用 set 一样啦！ 那么 declare 还有什么语法呢？看看先：

```
[root@linux ~]# declare [-aixr] variable
参数：
-a  ：将后面的 variable 定义成为数组 (array)
-i  ：将后面接的 variable 定义成为整数数字 (integer)
```

```
-x  : 用法与 export 一样，就是将后面的 variable 变成环境变量；
-r  : 将一个 variable 的变量设定成为 readonly ，该变量不可被更改内容，也不能 unset
范例：
范例一：让变量 sum 进行 100+300+50 的加总结果
[root@linux ~]# sum=100+300+50
[root@linux ~]# echo $sum
100+300+50  <==咦！怎么没有帮我计算加总？因为这是文字型态的变量属性啊！
[root@linux ~]# declare -i sum=100+300+50
[root@linux ~]# echo $sum
450         <==瞭乎？？

范例二：将 sum 变成环境变量
[root@linux ~]# declare -x sum

范例三：让 sum 变成只读属性，不可更动！
[root@linux ~]# declare -r sum
[root@linux ~]# sum=tesgting
-bash: sum: readonly variable  <==老天爷～不能改这个变数了！
```

declare 也是个很有用的功能～尤其是当我们需要使用到底下的数组功能时，他也可以帮我们宣告数组的属性喔！不过，老话一句，数组也是在 shell script 比较常用的啦！

- 数组属性 array 说明

某些时候，我们必须使用数组来宣告一些变量，这有什么好处啊？在一般人的使用上，果然是看不出来有什么好处的！不过，如果您曾经写过程序的话，那才会比较了解数组的意义～ 数组对写数值程序的设计师来说，可是不能错过学习的重点之一哩！好！不啰唆～ 那么要如何设定数组的变量与内容呢？在 bash 里头，数组的设定方式是：

　　var[index]=content

意思是说，我有一个数组名为 var ，而这个数组的内容为 var[1]=小明，var[2]=大明，var[3]=好明 .... 等等，那个 index 就是一些数字啦，重点是用中刮号 ([ ]) 来设定的。 目前我们 bash 提供的是一维数组。老实说，如果您不必写一些复杂的程序， 那么这个数组的地方，可以先略过，等到有需要再来学习即可！因为要制作出数组， 通常与循环或者其它判断式交互使用才有比较高的意义存在！

```
范例：设定上面提到的 var[1] ～ var[3] 的变数。
[root@linux ~]# var[1]="small min"
[root@linux ~]# var[2]="big min"
[root@linux ~]# var[3]="nice min"
[root@linux ~]# echo "${var[1]}, ${var[2]}, ${var[3]}"
```

比较有趣的地方在于『读取』，一般来说，建议直接以 ${数组} 的方式来读取， 比较正确无误的啦！

与档案系统及程序的限制关系： ulimit

想象一个状况：我的 Linux 主机里面同时登入了十个人，这十个人不知怎么搞的， 同时开启了 100 个档案，每个档案的大小约 10MBytes ，请问一下，我的 Linux 主机的内存要有多大才够？ 10*100*10 = 10000

MBytes ～～ 老天爷，这样，系统不挂点才有鬼哩！为了要预防这个情况的发生，所以， 我们的 bash 是可以『限制使用者的某些系统资源』的，包括可以开启的档案数量， 可以使用的 CPU 时间，可以使用的内存总量等等。如何设定？用 ulimit 吧！

```
[root@linux ~]# ulimit [-SHacdflmnpstuv] [配额]
参数：
-H  ：hard limit ，严格的设定，必定不能超过设定的值；
-S  ：soft limit ，警告的设定，可以超过这个设定值，但是会有警告讯息，
      并且，还是无法超过 hard limit 的喔！也就是说，假设我的 soft limit
      为 80 ， hard limit 为 100 ，那么我的某个资源可以用到 90 ，
      可以超过 80 ，还是无法超过 100 ，而且在 80~90 之间，会有警告讯息的意思。
-a  ：列出所有的限制额度；
-c  ：可建立的最大核心档案容量 (core files)
-d  ：程序数据可使用的最大容量
-f  ：此 shell 可以建立的最大档案容量 (一般可能设定为 2GB)单位为 Kbytes
-l  ：可用于锁定 (lock) 的内存量
-p  ：可用以管线处理 (pipe) 的数量
-t  ：可使用的最大 CPU 时间 (单位为秒)
-u  ：单一使用者可以使用的最大程序(process)数量。
范例：
范例一：列出所有的限制数据
[root@linux ~]# ulimit -a

范例二：限制使用者仅能建立 1MBytes 以下的容量的档案
[root@linux ~]# ulimit -f 1024
```

还记得我们在 Linux 磁盘档案系统 里面提到过，单一 filesystem 能够支持的单一档案大小与 block 的大小有关。例如 block size 为 1024 byte 时，单一档案可达 16GB 的容量。但是，我们可以用 ulimit 来限制使用者可以建立的档案大小喔！ 利用 ulimit -f 就可以来设定了！例如上面的范例二，要注意单位喔！单位是 Kbytes。 若改天你一直无法建立一个大容量的档案，记得瞧一瞧 ulimit 的信息喔！(不过，要注意的是，一般身份使用者如果以 ulimit 设定了 -f 的档案大小， 那么他『只能减小档案大小，不能增加档案大小喔！』)

额外的变量设定功能

刚刚我们提到了两种变量取用的方法，分别是这样：

```
[root@linux ~]# echo $HOME
[root@linux ~]# echo ${HOME}
```

那么，在那个 ${variable} 的使用方法中，其实，我们还可以将变量进行一些修订的工作喔！ 只要加上一些字符标志，后面再接着使用比对字符串，就能够修改变量的内容了！ 我们取底下的例子来说明：在底下的例子中，假设我的变量名称为 vbird ，且内容为 /home/vbird/testing/testing.x.sh。

```
1. 完整呈现 vbird 这个变量的内容；
[root@linux ~]# vbird="/home/vbird/testing/testing.x.sh"
[root@linux ~]# echo ${vbird}
```

```
/home/vbird/testing/testing.x.sh

2. 在 vbird 变量中，从最前面开始比对，若开头为 / ，则删除两个 /
   之间的所有数据，亦即 /*/
[root@linux ~]# echo ${vbird##/*/}
testing.x.sh    <==删除了 /home/vbird/testing/
[root@linux ~]# echo ${vbird#/*/}
vbird/testing/testing.x.sh   <==仅删除 /home/ 而已
# 这两个小例子有趣了～变量名称后面如果接了两个 ## ，表示在 ##
# 后面的字符串取『最长的』那一段；如果仅有一个 # ，表示取『最小的那一段』喔！

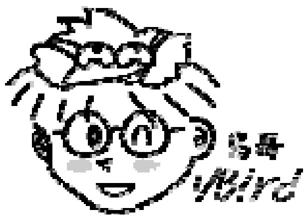
3. 承上题，如果是从后面开始，删除 /* 呢？
[root@linux ~]# echo ${vbird%%/*/}
/home/vbird/testing/testing.x.sh  <==都没被删除
[root@linux ~]# echo ${vbird%%/*}
    <==被删除光了！
[root@linux ~]# echo ${vbird%/*}
/home/vbird/testing   <==只删除 /testing.x.sh 部分
# 这个例子当中需要特别注意，那个 % 比对的是『最后面那个字符』的意思，
# 所以啰，第一个方式当然不对～因为 vbird 这个变量的内容最后面是 h 而不是 / 啊！
# 至于 %%/* 则是删除『最长的那个 /* 』，当然就是全部喔！而 %/* 则是最短的那个！

4. 将 vbird 变数中的 testing 取代为 TEST
[root@linux ~]# echo ${vbird/testing/TEST}
/home/vbird/TEST/testing.x.sh
[root@linux ~]# echo ${vbird//testing/TEST}
/home/vbird/TEST/TEST.x.sh
# 如果变量后面接的是 / 时，那么表示后面是进行『取代』的工作～而且仅取代『第一个』
# 但如果是 // ，则表示全部的字符串都取代啊！
```

这里您稍微留意一下就好了～反正就是变量后面可以接 #, ##, %, %%, /, // ， 而他们存在的意义并不相同的啦～

另外，几个不同的变量内容还可以进行判断呢！举例来说，目前我需要用到两个变量，分别是 var 与 str ，那我想要针对 str 这个变量内容是否有设定成一个字符串，亦即 "expr" 来决定 var 的内容。 那我可以使用什么方法来进行判断呢？如果您会写 shell script 的话， 直接用 shell script 就好了，如果不会写，那么我们就透过简单的变量判断吧！

Tips:
底下的例子当中，那个 var 与 str 为变量，我们想要针对 str 是否有设定来决定 var 的值喔！ 一般来说， str: 代表『str 没设定或为空的字符串时』；至于 str 则仅为『没有该变数』。



变量设定方式	str 没有设定	str 为空字符串	str 已设定非为空字符串

var=${str-expr}	var=expr	var=	var=$str
var=${str:-expr}	var=expr	var=expr	var=$str
var=${str+expr}	var=expr	var=expr	var=expr
var=${str:+expr}	var=expr	var=	var=expr
var=${str=expr}	str=expr var=expr	str 不变 var=	str 不变 var=$str
var=${str:=expr}	str=expr var=expr	str=expr var=expr	str 不变 var=$str
var=${str?expr}	expr 输出至 stderr	var=	var=str
var=${str:?expr}	expr 输出至 stderr	expr 输出至 stderr	var=str

根据上面这张表，我们来进行几个范例的练习吧！ ^_^

```
范例一：若 str 这个变量内容存在，则 var 设定为 str ，否则 var 设定为 "newvar"
[root@linux ~]# unset str; var=${str-newvar}
[root@linux ~]# echo var="$var", str="$str"
var=newvar, str=        <==因为 str 不存在，所以 var 为 newvar
[root@linux ~]# str="oldvar"; var=${str-newvar}
[root@linux ~]# echo var="$var", str="$str"
var=oldvar, str=oldvar  <==因为 str 存在，所以 var 等于 str 的内容

范例二：若 str 不存在，则 var 与 str 均设定为 newvar，否则仅 var 为 newvar
[root@linux ~]# unset str; var=${str=newvar}
[root@linux ~]# echo var="$var", str="$str"
var=newvar, str=newvar  <==因为 str 不存在，所以 var/str 均为 newvar
[root@linux ~]# str="oldvar"; var=${str=newvar}
[root@linux ~]# echo var="$var", str="$str"
var=oldvar, str=oldvar  <==因为 str 存在，所以 var 等于 str 的内容

范例三：若 str 这个变量存在，则 var 等于 str ，否则输出 "novar"
[root@linux ~]# unset str; var=${str?novar}
-bash: str: novar       <==因为 str 不存在，所以输出错误讯息
[root@linux ~]# str="oldvar"; var=${str?novar}
[root@linux ~]# echo var="$var", str="$str"
var=oldvar, str=oldvar  <==因为 str 存在，所以 var 等于 str 的内容

# 上面这三个案例都没有提到当 str 有设定，且为空字符串的情况喔！
# 您可以自行测试一下哩！
```

虽然猛一看，觉得变量没有什么奇特的地方，但是，如果仔细瞧一瞧，嘿！一堆环境变量与系统资源方面的变量， 可是会影响到我们在 bash 里头是否能够顺利作业的呢！例如 PATH 啊、ulimit 之类的～ 所以，您还是得要了解变量这个玩意才行喔！ ^_^

命令别名与历史命令：

我们知道在早期的 DOS 年代，清除屏幕上的信息可以使用 cls 来清除，但是在 Linux 里面， 我们则是使用 clear 来清除画面的。那么可否让 cls 等于 clear 呢？可以啊！用啥方法？ link file 还是什么的？别急！底下我们介绍不用 link file 的命令别名来达成。那么什么又是历史命令？ 曾经做过的举动我们可以将他记录下来喔！那就是历史命令啰～底下分别来谈一谈这两个玩意儿。

命令别名设定： alias， unalias

命令别名是一个很有趣的东西，特别是你的惯用指令特别长的时候！还有， 增设预设的属性在一些惯用的指令上面，可以预防一些不小心误杀档案的情况发生的时候！ 举个例子来说，如果你要查询隐藏档，并且需要长的列出与一页一页翻看，那么需要下达『 ls -al ｜ more 』这个指令，我是觉得很烦啦！ 要输入好几个单字！那可不可以使用 lm 来简化呢？！当然可以，你可以在命令列下面下达：

```
[root@linux ~]# alias lm=’ls -l | more’
```

嘿嘿！我立刻多出了一个可以执行的指令喔！这个指令名称为 lm ，且其实他是执行 ls -al ｜ more 啊！真是方便。不过， 要注意的是：『alias 的定义规则与变量定义规则几乎相同』， 所以你只要在 alias 后面加上你的 {『别名』=’指令 参数’ }， 以后你只要输入 lm 就相当于输入了 ls -al|more 这一串指令！很方便吧！

另外，命令别名的设定还可以取代既有的指令喔！举例来说，我们知道 root 可以移除(rm)任何数据！所以当你以 root 的身份在进行工作时，需要特别小心， 但是总有失手的时候，那么 rm 提供了一个参数来让我们确认是否要移除该档案，那就是 -i 这个参数！所以，你可以这样做：

```
[root@linux ~]# alias rm=’rm -i’
```

嘿嘿！那么以后使用 rm 的时候，就不用太担心会有错误删除的情况了！这也是命令别名的优点啰！ 那么如何知道目前有哪些的命令别名呢？就使用 alias 呀！

```
[root@linux ~]# alias
alias l.=’ls -d .* --color=tty’
alias ll=’ls -l --color=tty’
alias lm=’ls -al | more’
alias ls=’ls --color=tty’
alias vi=’vim’
alias which=’alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde’
```

由上面的资料当中，您也会发现一件事情啊，我们在 vi 文书编辑器 里面提到 vi 与 vim 是不太一样的，vi 是比较老，而 vim 可以用来取代 vi 喔。我们的 FC4 明明就同时有 vi/vim ， 为何我执行 vi 会是进入 vim 呢？呵呵！那就是因为上面的表格当中的『 alias vi=’vim’ 』这个设定啦！ 至于如果要取消命令别名的话，那么就使用 unalias 吧！例如要将刚刚的 lm 命令别名拿掉，就使用：

```
[root@linux ~]# unalias lm
```

那么命令别名与变量有什么不同呢？基本上，他们的意义就不太一样了！ alias 这种命令别名，你可以将他想成是建立一个新的指令名称， 至于变量则仅是将一个数值或者字符串存在某个代表意义当中！举个例

子好了，我们知道以前的 DOS 年代，列出目录与档案就是 dir ，而清除屏幕就是 cls ，那么如果我想要在 linux 里面也使用相同的指令呢？那就以 alias 来进行指令的别名设定：

```
alias cls='clear'
alias dir='ls -l'
```

只要加入这两行，以后你输入 cls 及 dir 就可以执行了！很方便吧！

历史命令：history

前面我们提过 bash 有提供指令历史的服务！那么如何查询我们曾经下达过的指令呢？就使用 history 啰！当然，如果觉得 histsory 要输入的字符太多太麻烦，可以使用命令别名来设定呢！ 不要跟我说还不会设定呦！ ^_^

```
alias h='history'
```

如此则输入 h 等于输入 history 啰！好了，我们来谈一谈 history 的用法吧！

```
[root@linux ~]# history [n]
[root@linux ~]# history [-c]
[root@linux ~]# history [-raw] histfiles
参数：
n   ：数字，意思是『要列出最近的 n 笔命令列表』的意思！
-c  ：将目前的 shell 中的所有 history 内容全部消除
-a  ：将目前新增的 history 指令新增入 histfiles 中，若没有加 histfiles ，
      则预设写入 ~/.bash_history
-r  ：将 histfiles 的内容读到目前这个 shell 的 history 记忆中；
-w  ：将目前的 history 记忆内容写入 histfiles 中！
范例：
范例一：列出目前内存内的所有 history 记忆
[root@linux ~]# history
# 前面省略
 1017  man bash
 1018  ll
 1019  history
 1020  history
# 列出的信息当中，共分两栏，第一栏为该指令在这个 shell 当中的代码，
# 另一个则是指令本身的内容喔！至于会秀出几笔指令记录，则与 HISTSIZE 有关！

范例二：列出目前最近的 3 笔资料
[root@linux ~]# history 3
 1019  history
 1020  history
 1021  history 3

范例三：立刻将目前的资料写入 histfile 当中
[root@linux ~]# history -w
# 在预设的情况下，会将历史纪录写入 ~/.bash_history 当中！
```

```
[root@linux ~]# echo $HISTSIZE
1000
```

在正常的情况下，当我们以 bash 登入 Linux 主机之后，系统会主动的由家目录的 ~/.bash_history 读取以前曾经下过的指令，那么 ~/.bash_history 会记录几笔数据呢？这就与你 bash 的 HISTSIZE 这个变量设定值有关了！在预设的 FC4 底下，是会记录 1000 笔数据的！ 那么假设我这次登入主机后，共下达过 100 次指令，『等我注销时， 系统就会将 101~1100 这总共 1000 笔历史命令更新到 ~/.bash_history 当中。』 也就是说，历史命令在我注销时，会将最近的 HISTSIZE 笔记录到我的纪录文件当中啦！ 当然，也可以用 history -w 强制立刻写入的！那为何用『更新』两个字呢？ 因为 ~/.bash_history 记录的笔数永远都是 HISTSIZE 那么多，旧的讯息会被主动的拿掉！ 仅保留最新的！

那么 history 这个历史命令只可以让我查询命令而已吗？呵呵！当然不止啊！ 我们可以利用相关的功能来帮我们执行命令呢！举例来说啰：

```
[root@linux ~]# !number
[root@linux ~]# !command
[root@linux ~]# !!
参数：
number  ：执行第几笔指令的意思；
command ：由最近的指令向前搜寻『指令串开头为 command』的那个指令，并执行；
!!      ：就是执行上一个指令(相当于按↑按键后，按 Enter)
范例：
[root@linux ~]# history
   66  man rm
   67  alias
   68  man history
   69  history
[root@linux ~]# !66  <==执行第 66 笔指令
[root@linux ~]# !!   <==执行上一个指令，本例中亦即 !66
[root@linux ~]# !al  <==执行最近以 al 为开头的指令(上头列出的第 67 个)
```

经过上面的介绍，瞭乎？历史命令用法可多了！如果我想要执行上一个指令， 除了使用上下键之外，我可以直接以『 !! 』 来下达上个指令的内容，此外， 我也可以直接选择下达第 n 个指令，『 !n 』来执行，也可以使用指令标头，例如 『 !vi 』来执行最近指令开头是 vi 的指令列！相当的方便而好用！基本上 history 的用途很大的！但是需要小心安全的问题！尤其是 root 的历史纪录档案，这是 Cracker 的最爱！因为不小心的 root 会将很多的重要数据在执行的过程中会被纪录在 ~/.bash_history 当中，如果这个档案被解析的话，后果不堪呐！无论如何，使用 history 配合『 ! 』曾经使用过的指令下达是很有效率的一个指令方法！

Bash Shell 使用环境：

是否记得我们登入主机的时候，屏幕上头会有一些说明文字，告知我们的 Linux 版本啊什么的， 还有，登入的时候，我们还可以给予使用者一些讯息或者欢迎文字呢。此外， 我们习惯的环境变量、命令别名等等的，是否可以登入就主动的帮我设定好？ 这些都是需要来注意的。另外，这些设定值又可以分为系统整体设定值与各人喜好设定值， 仅是一些档案放置的地点不同啦！这我们后面也会来谈一谈的！

绝对路径与相对路径

这个议题说到快要烂掉了～从一开始到现在，这个绝对路径与相对路径的问题我们就提到不知道多少次了，因为他实在很重要～这与 PATH 这个变量关系很大！老实说， 万一你的 PATH 没有设定完整的时候，下达指令就必须要以『 一长列的指令连带根目录都要列出来 』，呵呵那就是绝对路径的设定法啦！ 基本上，这个『 绝对路径』与『相对路径 』的观念是很重要的！否则你将常常会找不到档案说！ 所谓的『绝对路径』就是以根目录开始写入到档案的一种命令写定方法，举例来说，我目前在 /home/test 这个 test 使用者的家目录中，我想要看看里面的 .bashrc 这个档案的数据，使用的是 more 这个指令，而这个指令在 /bin/more 当中，则正确的下达指令的方法为：

```
[root@linux ~]# /bin/more .bashrc
```

我在的目录为 /home/test ！这是绝对路径写法！ 而如果你还记得我们在 Linux 档案与目录管理 那一篇文章中提到的观念的话，那么应该记得使用 ls -al 时会出现两个一定存在的目录，分别是『.』与『..』，分别代表是『这个路径』，与『上一层路径』！

```
[root@linux ~]# ls -al
drwxrwxr-x  2 root  root   4096  Aug 15 11:05 .
drwxrwxr-x  2 root  root   4096  Aug 14 23:26 ..
```

所以说，要执行上一层目录中的命令，可以下达『../command 』那个 command 指的是存在的可执行档！那么我因为在 /home/test 里面，距离 /bin 有两层上层目录，所以我要使用 /bin/more 这个执行档，并且使用相对路径的方法，就必须使用：

```
[root@linux ~]# ../../bin/more .bashrc
```

这种相对路径的方法相当广泛的被运用于 script 当中，这是因为如前面提到的， 每个人的安装预设的目录都不相同，则使用相对路径的话， 很容易就可以找到套件之间相依软件或者是设定档案的相关性！

例题：关于路径搜寻的问题！为何不执行目前所在目录下的档案？

答：

咦！刚刚不是提到『.』与『..』吗？那么那个『 . 』是干嘛用的？！眼尖的朋友应该已经发现了，就是『我在执行档案的时候，基本上，并不会主动搜寻目前目录下的档案』举个例子来说，我安装的 squid 这个执行档在 /usr/local/squid/bin/squid 这个档案，然而我在 /usr/local/squid/bin 下达 squid 的时候，系统会告诉你『查不到这个档案！』真是见鬼了！明明有这个档案的呀！这是因为系统预设的 PATH (路径) 并没有执行目前目录下的设定，也就是『.』这个路径！你可以使用『 echo $PATH 』看看，就可以知道为什么了！

那么为何不要设定这个路径呢？这是因为『 安全』的考虑。由于系统预设是允许任何人在 /tmp 底下写入任何档案的，那么万一有居心不良的使用者或者是 Cracker 入侵你的计算机，并在你的 /tmp 里头埋了一个小木马，并取名为 ls ，好了，改天你以 root 身份登入后，到 /tmp 底下，并执行 ls ，你看会有什么结果？！这个 /tmp/ls 由其它身份的人来执行或许没有问题，但是由 root 来执行却可能会导致 Cracker 所乐意见到的结果！那晓得为何了吧？！

> 当然啰！您还是可以选择在 ~/.bashrc 当中设定你的 . 在你的 PATH 当中，不过并不这么建议就是了！

好了，由于系统预设并不主动搜寻目前目录下的执行文件，那么你应该如何执行『目前目录下的执行文件』呢？ 很简单呀！就是以相对路径的观念，由于『 .. 』是上层，而『 . 』是这一层，所以要执行这一层目录的命令就使用『 ./command 』即可！例如你的 /usr/local/squid/bin 底下执行 squid 则可以写成：

```
[root@linux ~]# ./squid
```

请特别留意这方面的问题！『新手特别容易犯这个错误呢！』

登录讯息显示数据： /etc/issue, /etc/motd

还记得我们在终端机接口 (tty1 ~ tty6) 登入的时候，会有几行提示的字符串吗？ 那个字符串写在哪里啊？呵呵！在 /etc/issue 里面啊！先来看看：

```
[root@linux ~]# cat /etc/issue
Fedora Core release 4 (Stentz)
Kernel \r on an \m
```

在 FC4 里面预设有三行，这个在我们本机登入时就会显示在 title 的地方呢～ 咦！那么那个 \r 及 \m 是啥？您可以使用 man issue 配合 man mingetty 就能够知道：

issue 内的各代码意义
\d 本地端时间的日期；
\l 显示第几个终端机接口；
\m 显示硬件的等级 (i386/i486/i586/i686...)；
\n 显示主机的网络名称；
\o 显示 domain name；
\r 操作系统的版本 (相当于 uname -r)
\t 显示本地端时间的时间；
\s 操作系统的名称；
\v 操作系统的版本。

所以，如果您想要显示终端机的号码，就可以加上 \l 在 /etc/issue 档案内啰～就能够修改登入字符。咦！但是还有个 /etc/issue.net 呢！这是啥？没啥啦！这个是提供给 telnet 这个远程登入程序用的。当我们使用 telnet 连接到主机时，主机的登入画面就会显示 /etc/issue.net 而不是 /etc/issue 呢！

至于如果您想要让使用者登入后取得一些讯息，例如您想要让大家都知道的讯息， 那么可以将讯息加入 /etc/motd 里面去！例如：当登入后，告诉登入者， 系统将会在某个固定时间进行维护工作，可以这样做：

```
[root@linux ~]# vi /etc/motd
Hello everyone,
Our server will be maintained at 2005/10/10 0:00 ~ 24:00.
```

```
Please don't login at that time. ^_^
```

那么当你的使用者登入主机后，就会显示这样的讯息出来：

```
Last login: Mon Aug 15 10:17:10 2005 from 127.0.0.1
Hello everyone,
Our server will be maintained at 2005/10/10 0:00 ~ 24:00.
Please don't login at that time. ^_^
```

是否很方便啊！？ ^_^

环境设定档： bashrc, ~/.bashrc, ~/.profile, profile...,/etc/inputrc, source

关于取得 bash 的环境变量等数据，其实可以有系统规划与各人喜好， 一般来说，建议使用者直接修改个人设定值即可，不需要更动到系统啦～ 底下我们分别来谈一谈几个有趣的设定档喔！要注意的是，在指令列输入的变量也好、命令别名也罢， 都是针对该次登入的设定而已，所以只要您一注销，那么上次的设定值就会不见去！ 因此，我们需要有几个档案来帮助我们，每次登入的时候，就已经帮我们搞定了环境的设定啰！

- 系统设定值

所谓的系统设定值，也就是说每个使用者进入到 bash shell 之后，会先读取的设定档案！ 预设的设定档案有下列几个：

- /etc/sysconfig/i18n

记得我们在几个重要变量内谈到的语系数据吗？！ 那个语系是由 i18n 所维护的，而 FC4 预设的系统语系设定文件就在 /etc/sysconfig/i18n 当中。 这个档案有点像这样：

```
[root@linux ~]# cat /etc/sysconfig/i18n
LANG="zh_TW.UTF-8"
SYSFONT="latarcyrheb-sun16"
SUPPORTED="zh_TW.UTF-8:zh_TW:zh:en_US.UTF-8"
```

我预设使用 zh_TW.UTF-8 来作为我的整体语系，当然，我可以在这里修改 LANG 以及其它相关的语系变量，例如 LC_CTYPE 或者是 LC_TIME 等等的。不过，一般来说，使用者自己个人的设定不建议在这里做更动啦！他们可以自行设定他们自己的设定档啊！

- /etc/profile

这个档案设定了几个重要的变量，例如：『PATH、USER、MAIL、 HOSTNAME、HISTSIZE、umask』等等，也同时规划出 /etc/inputrc 这个针对键盘热建设定的档案的数据内容。你可以在这里设定总体的 PATH 等等的信息！ 同时，这个档案也规划出 /etc/profile.d 及 /etc/inputrc 这两个目录与档案！

总之，你可以了解到刚刚我们学会的变量设定方式，在这个档案中也可以设定呢！ 但是设定上需要特别小心，因为所有的使用者皆会使用到这个档案的信息。通常我都喜欢将 /usr/local/bin 这个路径加成最前面，这是因为通常自己安装的套件自己最喜欢， 所以当然是最先搜寻啰！ ^_^！此外，请注意一下，可以将 HISTSIZE 的大小改变一下，改成 50 就可以啦！比较安全！(注：这个档案不论在那个 Linux distributions 当中均存在 /etc/profile 当中，所以，请特别留意此一档案即可！)。

- /etc/bashrc

这个档案在规划 umask 的功能，也同时规划出提示字符的内容 (就是里头那个 PS1 啦！) 。特别留意的是，这个档案在不同的 Linux distribution 里面，摆放的位置可能不太一样呢！ 所以需要查询一下才行呦！

- /etc/profile.d/*.sh

/etc/profile.d 是一个目录，里面针对 bash 及 C-shell 规范了一些数据。 以 FC4 为例，这个目录里面就针对了颜色、语系、vim 及 which 等指令进行一些额外的设定， 例如 alias 之类的规范值。我们的 vim 被用 alias 命名为 vi 就是在这个目录下被设定好的。 当然啦，这个目录的由来其实是在 /etc/profile 这个档案内规范的啦！ 你可以自行设定一些 *.sh 的文件名的档案来书写自己的系统设定值喔！

- /etc/man.config

这个档案乍看之下好像跟 bash shell 没相关性，但是对于系统管理员来说， 却也是很重要的一个档案！这的档案的内容『规范了使用 man 的时候， man page 的路径到哪里去寻找！』所以说的简单一点，这个档案规定了下达 man 的时候，该去哪里查看数据的路径设定！那么什么时候要来修改这个档案呢？如果你是以 tarball 的方式来安装你的数据，那么你的 man page（指令说明档案）可能会放置在 /usr/local/softpackage/man 里头，那个 softpackage 是你的套件名称， 这个时候你就得以手动的方式将该路径加到 /etc/man.config 里头，否则使用 man 的时候就会找不到相关的说明档啰。

事实上，这个档案内最重要的其实是 MANPATH 这个变量设定啦！ 我们搜寻 man page 时，会依据 MANPATH 的路径去分别搜寻啊！另外，要注意的是， 这个档案在各大不同版本 Linux distributions 中，檔名都不太相同，例如 FC4 用的是 /etc/man.config ，而 SuSE 用的则是 /etc/manpath.config ， 可以利用 [tab] 按键来进行文件名的补齐啦！

这就是系统在设定的时候常常会使用的档案！需要特别留意的是，通常设定完了这几个档案之后，都需要先 logout 在 login 之后才会将设定整个启动起来！

- 个人设定值

那么个人的喜好设定在哪里？嘿嘿嘿嘿！那就是在个人家目录的几个隐藏文件当中啰！ 分别会使用到底下的几个档案啦！(注意！底下的档案都是隐藏档，需要使用 ls -al 方能显示出来) ，另外，注意一下啰！底下那个『 ~ 』代表的是『家目录』的意思：

- ~/.bash_profile, ~/.bash_login, ~/.profile

这三个档案通常只要一个就够了，一般预设是以 ~/.bash_profile 的檔名存在。 会有这么多的档案，其实是因应其它 shell 转换过来的使用者的习惯而已。 这个档案可以定义个人化的路径 (PATH) 与环境变量等等。不过，还是有顺位上的差异， bash 启动时，会先去读取 ~/.bash_profile，找不到时，就去读取 ~/.bash_login ，然后才是 ~/.profile。

- ~/.bashrc

鸟哥一般都是将自己的需要输入在这个档案里面的呢！ 我的个人化设定值都会写在这里说～例如命令别名、路径等等。 这个档案在您每次执行 shell script 的时候都会被重新使用一遍，所以是最完整的。 而上头的 ~/.bash_profile 则只有在登入的时候会被读取一次。

- ~/.bash_history

还记得我们在历史命令提到过这个档案吧？！呵呵！没错～预设的情况下， 我们的历史命令就记录在这里啊！而这个档案能够记录几笔数据，则与 HISTSIZE 这个变数有关啊。每次登入 bash 后，bash 会先读取这个档案，将所有的历史指令读入内存， 因此，当我们登入 bash 后就可以查知上次使用过哪些指令啰。至于更多的历史指令， 请自行回去参考喔！

- ~/.bash_logout

这个档案则记录了『当我注销 bash 后，系统再帮我做完什么动作后才离开』的意思。 你可以去读取一下这个档案的内容，预设的情况下，注销时， bash 只是帮我们清掉屏幕的讯息而已。 不过，你也可以将一些备份或者是其它你认为重要的工作写在这个档案中（例如清空暂存盘）， 那么当你离开 Linux 的时候，就可以解决一些烦人的事情啰！

好了，我们知道在变量的设定规范当中，后输入的设定值可以取代先输入的设定值， 那么在我们登入 bash 的时候，这些设定档到底是如何读取的呢？他是这样读取的：

1. 先读取 /etc/profile ，再根据 /etc/profile 的内容去读取其它额外的设定档， 例如 /etc/profile.d 与 /etc/inputrc 等等设定档；
2. 根据不同的使用者，到使用者家目录去读取 ~/.bash_profile 或 ~/.bash_login 或 ~/.profile 等设定档；
3. 根据不同使用者，到他家目录去读取 ~/.bashrc 。

所以啰，当我登入 bash 后，最终读取的设定档竟然是 ~/.bashrc 呢！ 也就是说，在 ~/.bashrc 里面的设定会是最终的设定值！所以啰， 通常鸟哥我喜欢将个人的一些常用 alias 或 PATH 等环境变量或自订变量都写到这个档案去， 如此一来，不论原来系统帮我们做了什么设定值，我都可以使用属于自己熟悉的环境呢！ 鸟哥的 ~/.bashrc 有点像这样：

```
[root@linux ~]# vi ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
PATH="/bin:/sbin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin"
PATH="$PATH":/usr/X11R6/bin:/home/dmtsai/bin
LANG=zh_TW.big5
LC_TIME=C
export PATH LC_TIME LANG
umask 022

alias   rm='rm -i'
alias   cp='cp -i'
alias   mv='mv -i'
alias   ll='ls -l'
```

```
alias    lm='ls -al|more'
alias    h='history'
```

仔细看到上头这个档案,会不会觉得奇怪啊!为什么会有第五行的『 . /etc/bashrc 』呢? 那个小数点 (.) 代表什么意思啊? ? 其实 if [...]; then .... fi 是 shell script 当中的程序写法, 这个我们会在下一章当中介绍。不过,那个 . 则需要好好的谈一谈喔!一般来说,如果修改完了设定档,通常就是 logout 后再重新 login 到 bash 内,就能够将环境设定档重读了!不过, 我们可以使用底下的方式来让该设定档立即生效:

```
[root@linux ~]# source file
范例:
[root@linux ~]# source ~/.bashrc
[root@linux ~]#  .  ~/.bashrc
```

利用 source 或小数点 (.) 都可以将设定档的内容读进来目前的 shell 环境中! 举例来说,我修改了 ~/.bashrc ,那么不需要注销,立即以 source ~/.bashrc 就可以将刚刚最新设定的内容读进来目前的环境中!很不错吧!此外,什么时候会使用到不同的设定档呢? 最常发生在一个人的工作环境分为多重的时候了!举个例子来说,在我的大型主机中, 我常常需要负责两到三个不同的案子,每个案子所需要处理的环境变量订定并不相同, 那么我就将这两三个案子分别编写属于该案子的环境变量设定档案,当我需要该环境时,就直接『 source 变量文件 』,如此一来,环境变量的设定就变的更简便而灵活了!

- login shell 与 non-login shell

事实上,这些环境设定档在读取时,还是有一些差异的,这就得要谈到所谓的『login shell』与『non-login shell』的差异了。基本上,就字面上的意义来解释的话,所谓的 loign shell 指的就是当使用者登入 Linux 系统时,所取得的那个 shell 称为 login shell。 当登入后,再去执行其它的 shell 时,其它的 shell 就是 non-login shell 了。举例来说,我以 dmtsai 这个使用者身份登入 Linux 后,然后为了要执行一些数值模拟的工作,而去执行 csh 这个 C shell , 那么此时我就取得了 non-login shell 了。

另外一个例子是,当我以 X Window 的环境登入 Linux 时,我们不是可以使用『终端机』来开启 shell 吗? 当登入 Linux 的时候所取得的那个 X 的环境也可以读入 login shell 的。因此,在 X 环境下所启动的终端机,那些 shell 都是 non-login shell 喔!

login 与 non-login shell 的差异除了取得的时机不同之外,其实他们读取的环境设定档也不相同。 我们上头说过一些个人的环境设定档案了吧?那么这两种类型的 shell 该读取什么档案呢? 当登入 Linux ,亦即是取得 login shell 时,会读取 ~/.bash_profile, ~/.bash_login, ~/.profile, 这三个档案的优先级已经在上面提过,自行参考一下。至于在取得 login shell 后继续动作的其它 non-login shell ,读取的就是仅有 ~/.bashrc 啰~。而大部分的 linux distributions 都会将 ~/.bash_profile 的内容指到 ~/.bashrc 去,这样比较简单啰~

终端机的环境设定: stty, set

什么叫做『终端机环境』啊? ! 我们在 首次登入 Linux 时就提过,可以在 tty1 ~ tty6 这六个文字接口的终端机 (terminal) 环境中登入,那么登入的时候我们可以取得一些字符设定的功能喔! 举例来说,我们可以利用退格键 (backspace,就是那个←符号的按键) 来删除命令列上的字符, 也可以使用 [ctrl]+c 来强制终止一个指令的运行,当输入错误时,就会有声音跑出来警告。这是怎么办到的呢? 很简单啊!因为登入终端机的时候,会自动的取得一些终端机的输入环境的设定啊!

事实上，目前我们使用的 Linux distributions 都帮我们作了最棒的使用者环境了， 所以大家可以不用担心操作环境的问题。不过，在某些 Unix like 的机器中，还是可能需要动用一些手脚， 才能够让我们的输入比较快乐～举例来说，利用 [backspace] 删除，要比利用 [Del] 按键来的顺手吧！ 但是某些 Unix 偏偏是以 [del] 来进行字符的删除啊！所以，这个时候就可以动动手脚啰～

那么如何查阅目前的一些按键内容呢？可以利用 stty (setting tty 终端机的意思) 呢！ stty 也可以帮助设定终端机的输入按键代表意义喔！

```
[root@linux ~]# stty [-a]
参数：
-a  ：将目前所有的 stty 参数列出来；
范例：
范例一：列出所有的按键与按键内容
[root@linux ~]# stty -a
speed 38400 baud; rows 40; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = ;
eol2 = ; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase
= ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl
ixon -ixoff -iuclc -ixany -imaxbel opost -olcuc -ocrnl onlcr -onocr
-onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 isig icanon iexten
echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl echoke
```

我们可以利用 stty -a 来列出目前环境中所有的按键列表，在上头的列表当中，需要注意的是特殊字体那几个， 此外，如果出现 ^ 表示 [Ctrl] 那个按键的意思。举例来说， intr = ^C 表示利用 [ctrl] + c 来达成的。 几个重要的代表意义是：

- eof : End of file 的意思，代表『结束输入』。
- erase : 向后删除字符，
- intr : 送出一个 interrupt (中断) 的讯号给目前正在 run 的程序；
- kill : 删除在目前指令列上的所有文字；
- quit : 送出一个 quit 的讯号给目前正在 run 的程序；
- start : 在某个程序停止后，重新启动他的 output
- stop : 停止目前屏幕的输出；
- susp : 送出一个 terminal stop 的讯号给正在 run 的程序。

记不记得我们讲过 Linux 底下的几个热键 啊？没错！ 就是这个 stty 设定值内的 intr / eof 啰～至于删除字符，就是 erase 那个设定值啦！ 如果你想要用 [ctrl]+h 来进行字符的删除，那么可以下达：

```
[root@linux ~]# stty erase ^h
```

那么从此之后，你的删除字符就得要使用 [ctrl]+h 啰，按下 [backspace] 则会出现 ^? 字样呢！ 如果想要回复利用 [backspace] ，就下达 stty erase ^? 即可啊！ 至于更多的 stty 说明，记得参考一下 man stty 的内容喔！

除了 stty 之外，其实我们的 bash 还有自己的一些终端机设定值呢！那就是利用 set 来设定的！ 我们之前提到一些变量时，可以利用 set 来显示，除此之外，其实 set 还可以帮我们设定整个指令输出/输入的环境。 例如记录历史命令、显示错误内容等等。

```
[root@linux ~]# set [-uvCHhmBx]
参数:
-u  : 预设不启用。若启用后，当使用未设定变量时，会显示错误讯息;
-v  : 预设不启用。若启用后，在讯息被输出前，会先显示讯息的原始内容;
-x  : 预设不启用。若启用后，在指令被执行前，会显示指令内容(前面有 ++ 符号)
-h  : 预设启用。与历史命令有关(下节介绍);
-H  : 预设启用。与历史命令有关(下节介绍);
-m  : 预设启用。与工作管理有关(未来介绍);
-B  : 预设启用。与刮号 [] 的作用有关;
-C  : 预设不启用。若使用 >  等，则若档案存在时，该档案不会被覆盖。
范例:
范例一: 显示目前所有的 set 设定值
[root@linux ~]# echo $-
himBH
# 那个 $- 变量内容就是 set 的所有设定啦！ bash 预设是 himBH 喔！

范例二: 设定 "若使用未定义变量时，则显示错误讯息"
[root@linux ~]# set -u
[root@linux ~]# echo $vbirding
-bash: vbirding: unbound variable
# 预设情况下，未设定/未宣告 的变量都会是『空的』，不过，若设定 -u 参数，
# 那么当使用未设定的变量时，就会有问题啦！很多的 shell 都预设启用 -u 参数。
# 若要取消这个参数，输入 set +u 即可！

范例三: 执行前，显示该指令内容。
[root@linux ~]# set -x
[root@linux ~]# echo $HOME
+ echo /root
/root
++ echo -ne '\033]0;root@linux:~\007'
# 看见否? 要输出的指令都会先被打印到屏幕上喔！前面会多出 + 的符号！
```

另外，其实我们还有其它的按键设定功能呢！就是在 /etc/inputrc 这个档案里面设定。

```
[root@linux ~]# cat /etc/inputrc
# do not bell on tab-completion
#set bell-style none

set meta-flag on
set input-meta on
set convert-meta off
set output-meta on
```

```
.....以下省略.....
```

还有例如 /etc/DIR_COLORS* 与 /etc/termcap 等，也都是与终端机有关的环境设定档案呢！ 不过，事实上，鸟哥并不建议您修改 tty 的环境呢，这是因为 bash 的环境已经设定的很亲和了， 我们不需要额外的设定或者修改，否则反而会产生一些困扰。不过，写在这里的数据， 只是希望大家能够清楚的知道我们的终端机是如何进行设定的喔！ ^_^

万用字符与特殊符号：

嘿嘿！在 bash 里头还支持一些万用字符喔 (wild card) ！多了这些万用字符， 我们利用 bash 处理数据就更方便了！底下我们列出一些常用的万用字符喔：

符号	内容
*	万用字符，代表 0 个或多个字符（或数字）
?	万用字符，代表『一定有』一个字母
#	批注，这个最常被使用在 script 当中，视为说明！
\	跳脱符号，将『特殊字符或万用字符』还原成一般字符
		分隔两个管线命令的界定；
;	连续性命令的界定（注意！与管线命令并不相同）
~	使用者的家目录
$	亦即是变量之前需要加的变量取代值
&	将指令变成背景下工作
!	逻辑运算意义上的『非』 not 的意思！
/	路径分隔的符号
>, >>	输出导向，分别是『取代』与『累加』
’	单引号，不具有变量置换的功能
”	具有变量置换的功能！
` `	两个『 ` 』中间为可以先执行的指令！
( )	在中间为子 shell 的起始与结束
[ ]	在中间为字符的组合
{ }	在中间为命令区块的组合！
组合按键	**执行结果**
Ctrl + C	终止目前的命令
Ctrl + D	输入结束（EOF），例如邮件结束的时候；
Ctrl + M	就是 Enter 啦！

Ctrl + S	暂停屏幕的输出
Ctrl + Q	恢复屏幕的输出
Ctrl + U	在提示字符下，将整列命令删除
Ctrl + Z	『暂停』目前的命令

在上面的『按键组合』当中，有没有发现跟上个小节很相似的内容啊！？ 呵呵～没错啦！那些组合键都可以在 stty 当中来进行不同的设定的！好玩吧！ 至于上面的万用字符当中，最常用的就属 *, ?, [] 及 ` 了！我们提几个简单的例子：

```
[root@linux ~]# ls test*      <==那个 * 代表后面不论接几个字符都予以接受
[root@linux ~]# ls test?      <==那个 ? 代表后面『一定』要接『一个』字符
[root@linux ~]# ls test???    <==那个 ??? 代表『一定要接三个』字符！
[root@linux ~]# cp test[1-5] /tmp
# 将 test1, test2, test3, test4, test5 若存在的话，就拷贝到 /tmp
[root@linux ~]# cp test[!1-5] /tmp
# 只要不是 test1, test2, test3, test4, test5 之外的其它 test? ,
# 若存在的话，就拷贝到 /tmp
[root@linux ~]# cd /lib/modules/`uname -r`/kernel/drivers
# 被 ` ` 括起来的内容『会先执行』
```

上面几个例子相当的有趣！尤其是最后面两个！需要注意的是， [1-5] 里面『代表只有一个字符』但是范围可以由 1-5 ，这样来说的话，那么我们如果允许『只要档名里面含有至少一个大写字符』时，就可以将档案 copy 出来的话，可以这样做：

```
cp *[A-Z]* /tmp
```

很有趣吧？！也就是说『 [ ] 谨代表一个字符，而这个字符的定义可以是范围(-)， 可以是指定项目，也可以是两者并存。 』举例来说，我想要找出在 /etc/ 底下所有含有数字的档案， 可以这样：

```
ls -lda /etc/*[0-9]*
```

但如果我只想要找出含有 3 及 5 的档名的档案呢？就会是这样：

```
ls -lda /etc/*[35]*
```

如果是『不想要』某些范围或者是单字呢？就使用 [!] 即可！例如不想要有小写字符为开头的档案：

```
ls -lda /etc/[!a-z]*
```

很好玩吧！至于那个 ` 是啥？在一串指令当中， `command` 内的指令会先被执行， 执行完的讯息再回传到外部指令来处理！也就是说：

1. 系统先执行 uname -r 找出输出的结果；
2. 将结果累加在目录上面，来执行 cd 的功能！

很棒吧！！另外，这个 quot (`) 的功能，也可以利用 $() 来取代喔！例如：

```
cd /lib/modules/$(uname -r)/kernel
```

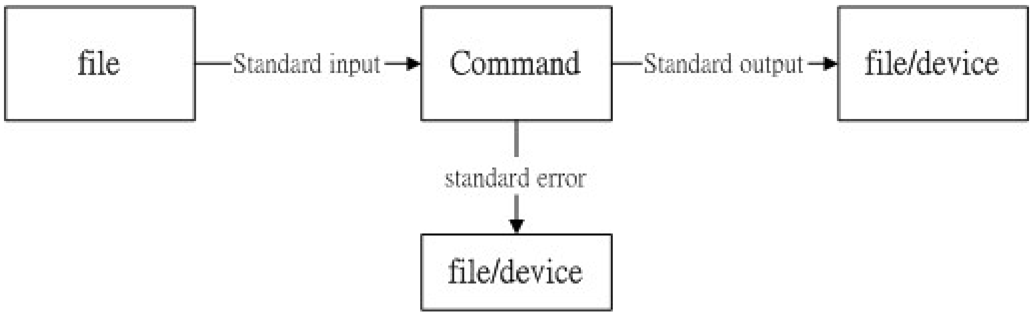
这些基本的功能需要特别来了解一下才行呦！至于更多的使用方式， 我们会在后续的正规表示法当中在详谈的！

数据流重导向

数据流重导向 (redirect) 由字面上的意思来看，好像就是将『数据给他传导到其它地方去』的样子？ 呵呵！是啊是啊！没错～数据流重导向就是将某个指令执行后应该要出现在屏幕上的数据， 给他传输到其它的地方，例如档案或者是装置 (例如打印机之类的！)！这玩意儿在 Linux 的文字模式底下可重要的！ 尤其是如果我们想要将某些数据储存下来时，就更有用了！

什么是数据流重导向

好家伙！什么是数据流重导向啊？这得要由指令的执行结果谈起！ 一般来说，如果你要执行一个指令，通常他会是这样的：



图三、指令执行过程的数据传输情况

我们执行一个指令的时候，这个指令可能会由档案读入资料，经过处理之后，再将数据输出到屏幕上。 在图三当中， standard output 与 standard error 分别代表标准输出与标准错误输出， 这两个玩意儿预设都是输出到屏幕上面来的啊！举个简单例子来说， 我们下达『 cat /etc/crontab /etc/vbirdsay 』这个指令时，cat 会由 /etc/crontab 与 /etc/vbirdsay 读入数据， 然后再将数据输出到屏幕上，不过，因为系统本来就不存在 /etc/vbirdsay 这个档案， 所以就会显示错误讯息，这个错误讯息也会输出到屏幕上来喔！

在这样的过程当中，我们可以将 standard error (简称 stderr) 与 standard output (简称 stdout) 给他传送到其它不同的地方，而不是屏幕上头！传送的目标处，通常是档案或者是装置！ 而传送的指令则是如下所示：

1. 标准输入(stdin) ：代码为 0 ，使用 < 或 << ；
2. 标准输出(stdout)：代码为 1 ，使用 > 或 >> ；
3. 标准错误输出(stderr)：代码为 2 ，使用 2> 或 2>> ；

举例来说，如果我想要将我目前根目录下所有的目录都记录下来的话，也就是说，将 ls -l / 这个指令的输出结果储存下来，就可以：

```
[root@linux ~]# ls -l /  >  ~/rootfile
# 本来 ls -l / 会将根目录的数据列出到屏幕上；
# 现在我使用了 > ~/rootfile 后，则本来应该在屏幕上出现的数据
# 就会被『重新导向』到 ~/rootfile 档案内了！就可以将该数据储存！
```

此时，原本应该在屏幕上面出现的数据通通不见去～因为那些资料都被写入到 ~/rootfile 去了！ 当然，那个档案的档名随便你取啦～如果你下达：『 cat ~/rootfile 』就可以看到原本应该在屏幕上面的数据

啰。 那么如果我再次下达：『 ls -l /home > ~/rootfile 』后，那么那个 ~/rootfile 档案的内容变成什么？ 呵呵！变成『仅有 ls -l /home 的数据』而已！咦！原本的 ls -l / 数据就不见了吗？是的！ 因为该档案的建立方式是：

1. 该档案 (本例中是 ~/rootfile) 若不存在，系统会自动的将他建立起来，但是，
2. 当这个档案存在的时候，那么系统就会先将这个档案内容清空，然后再将数据写入！
3. 也就是若以 > 输出到一个既存盘案中，呵呵，那个档案就会被覆盖掉啰！

那如果我想要将数据累加，不想要将旧的数据删除，那该如何是好？ 呵呵！就利用 >> 就好啦！例如上面的例子中，就变成『ls -l / >> ~/rootfile』 如此一来，当 ~/rootfile 不存在时，系统会主动建立这个档案，若该档案已存在， 则数据会在该档案的最下方累加进去！基本上，指令的下达方式：

command	> 1> 2> 2>> <	装置或档案

当然啦，一串指令的最左边一定是指令，而在 >,2>,< 右边的，必须是档案或装置才行！ 此外，那个 > 会等于 1> ，因为 standard output 代码是 1 ，可以省略啦！ 再者， 1 与 > 之间并没有空格喔！是紧接在一起的！注意注意！我们底下来玩几个东西好了：

```
范例一：将目前目录下的档案信息全部储存到 list.txt 档案中
[root@linux ~]# ls -al > list.txt

范例二：将根目录下的数据也储存到 list.txt 档案中
[root@linux ~]# ls -al / >> list.txt
```

好了，对于『 >, >> 』这两个东西有一定的概念之后，我们来深入的谈一谈『数据流重导向』的观念吧！ 如前所述，基本上， Linux 执行的结果中，可以约略的分成『正确输出』与『错误输出』两种数据。 例如，当你以一般身份执行 find 这个指令时，例如执行『 find / -name testing 』时，由于你是一般身份，又有些数据夹是不允许一般身份者进入的， 所以啰，当你使用 find 时，就会有错误讯息发生了！但同时如果有 testing 这个档案在你可以进入的资料夹当中，那么屏幕也会输出到给你看！因此， 就具有正确的与错误的输出两种啰！ (分别称为 Stdout 与 Stderror) 例如下面为执行结果： 里面的『 find: /home/root: Permission denied 』就告诉你该数据夹你没有权限进入， 这就是错误的输出了，那么『 /home/dmtsai/tseting 』就是正确的输出了！

```
[dmtsai@linux ~]$ find /home -name testing
find: /home/test1: Permission denied    <== Starndard error
find: /home/root: Permission denied     <== Starndard error
find: /home/masda: Permission denied    <== Starndard error
/home/dmtsai/testing                    <== Starndard output
```

好了，那么假如我们想要将数据输出到 list 这个档案中呢？执行『 find / -name testing > list 』 会有什么结果？呵呵，你会发现 list 里面存了刚刚那个『正确』的输出数据， 至于屏幕上还是会有错误的讯息出现呢！伤脑筋！如果想要将正确的与错误的数据分别存入不同的档案中需要怎么做？！ 呵呵！其实

在数据的重导向方面，正确的写法应该是『 1> 』与『 2> 』才对！但是如果只有 > 则预设是以 1> 来进行数据的！那个 1> 是输出正确数据， 2> 则是错误数据输出项目。也就是说：

- 1> ：是将正确的数据输出到指定的地方去
- 2> ：是将错误的数据输出到指定的地方去

好了，那么上面的例子中，我们如何将数据输出到不同的地方去呢？可以这么写：

```
[dmtsai@linux ~]$ find /home -name testing > list_right 2> list_error
```

这样一来，刚刚执行的结果中，有 Permission 的那几行错误信息都会跑到 list_error 这个档案中，至于正确的输出数据则会存到 list_right 这个档案中啰！这样可以了解了吗？ 如果有点混乱的话，去休息一下再来看看吧！！

再来，如果我只要正确的数据，错误的信息我不要了呢？呵呵，这个时候 /dev/null 这个垃圾桶就很重要了！/dev/null 是什么呢？ 基本上，那就有点像是一个『黑洞』的垃圾桶功能！当你输入的任何东西导向到这个虚拟的垃圾桶装置时， 『他就会凭空消失不见了～～』，这个东西有用的很！例如上面的例子中，我们可以这么做，来将错误的信息丢掉！

```
[dmtsai@linux ~]$ find /home -name testing > list_right 2> /dev/null
```

很神奇呦！ error message 就会『不见了！』呵呵！真高兴！另外， 如果我要将数据都写到同一个档案中呢？这个时候写法需要用到特殊写法，请注意底下的写法呦！

```
[dmtsai@linux ~]$ find /home -name testing > list 2> list  <==错误写法
[dmtsai@linux ~]$ find /home -name testing > list 2>&1     <==正确写法
```

请特别留意这一点呢！同时写入同一个档案需要使用 2>&1 才对呦！

OK！了解了 >, 2>, >> 与 /dev/null 之后，那么那个 < 又是什么呀！？呵呵！以最简单的说法来说， 那就是『将原本需要由键盘输入的数据，经由档案来读入』的意思。 举例来说，我们可以使用 cat 在键盘上面输入一些数据，然后写入一个档案内，例如：

```
[root@linux ~]# cat > catfile
testing
cat file test
<==这里按下 [ctrl]+d 结束输入来离开！
```

此时就会有 catfile 这个档案产生，而且该档案的内容就是刚刚输入的内容喔。 那么，我是否可以使用其它档案来取代键盘输入呢？可以啊！这样做！

```
[root@linux ~]# cat > catfile < somefile
```

我可以先编辑 somefile ，然后再以上述的指令来将数据输出到 catfile 去呢！这样可以理解了吗？ 能够理解 < 之后，再来则是怪可怕一把的 << 这个连续两个小于的符号了～ 他代表的是『结束的输入字符』的意思！举例来讲：『我要用 cat 直接将输入的讯息输出到 catfile 中， 且当输入 eof 时，该次输入就结束』，那我可以这样做：

```
[root@linux ~]# cat > catfile <<eof
> This is a test testing
```

```
> OK now stop
> eof  <==输入这个玩意儿，嘿！立刻就结束了！
```

看到了吗？利用 << 右侧的控制字符，我们可以终止一次输入， 而不必输入 [crtl]+d 来结束哩！这对程序写作很有帮助喔！好了，那么为何要使用命令输出重导向呢？ 这个问题一定会困扰你一下下的，如果你从来都没有写过 script 的话！好了，我们来说一说吧！

- 当屏幕输出的信息很重要，而且我们需要将他存下来的时候；
- 背景执行中的程序，不希望他干扰屏幕正常的输出结果时；
- 一些系统的例行命令（例如写在 /etc/crontab 中的档案）的执行结果，希望他可以存下来时；
- 一些执行命令，我们已经知道他可能的错误讯息，所以想以『 2> /dev/null 』将他丢掉时；
- 错误讯息与正确讯息需要分别输出时。

当然还有很多很多的功能的，最简单的就是网友们常常问到的：『 为何我的 root 都会收到系统 crontab 寄来的错误讯息呢』这个咚咚是常见的错误， 而如果我们已经知道这个错误讯息是可以忽略的时候，嗯！『 2> errorfile 』这个功能就很重要了吧！ 了解了吗？？

命令执行的判断依据： ; , &&, ||

在某些时候，我们希望可以一次执行多个指令，例如关机时，希望我可以先执行两次 sync ，然后才 shutdown 计算机，那么可以怎么作呢？这样做呀：

```
[root@linux ~]# sync; sync; shutdown -h now
```

在指令与指令中间利用分号 (;) 来隔开，这样一来，分号前的指令执行完后， 就会立刻接着执行后面的指令了。这真是方便啊～再来，换个角度来想， 万一我想要在某个目录底下建立一个档案，也就是说，如果该目录存在的话， 那我才建立这个档案，如果不存在，那就算了～目录是否存在可以使用一些 bash 提供的判断式功能， 但这里假设我不晓得那个指令，但我知道我可以使用 ls 来判断是否有该目录的存在，也就是说，我可以利用 ls directoryname 判断是否存在，然后以 touch 建立一个档案， 这两个指令有相关性，那该如何写呢？呵呵！可以利用 && 来作喔！

```
[root@linux ~]# ls /tmp && touch /tmp/testingagin
```

是否记得我们在变量的章节里面谈过这个奇怪的变数『 $? 』呢？ 如果指令执行结果没有错误讯息，那就会回传 $?=0 ，如果有错误， 那回传值就不会是 0 啊！经由这样的判断，我们也可以利用 && 来决定， 当前面的指令执行结果为正确 (例如：仅有 standard output 时)，就可以接着执行后续的指令， 否则就予以略过！因此，当 ls /tmp 没有问题，那么就会接着执行 touch /tmp/testingagin 了！ 万一是这样：

```
[root@linux ~]# ls /vbird && touch /vbird/test
```

因为我的系统里面根本就不可能存在 /vbird 这个目录呢！所以，执行 ls /vbird 就会回传错误， 那么后续的 touch /vbird/test 自然就不会动作啰！了解吗？

再换个角度来想，如果我想要当某个档案不存在时，就去建立那个档案， 否则就略过呢？很简单啊～可以这样做：

```
[root@linux ~]# ls /tmp/vbirding || touch /tmp/vbirding
```

那个 || 刚好完全跟 && 相反，当前一个指令有错误时，在 || 后面的指令才会被执行！ (要注意，那个 | 是两个 | ，而 | 按键则是反斜线 \ 同一个按键， 因此，按下 [Shift] 加上 [\] 就会出现那个 | 啰！) 因此，简单的来说，当 ls /tmp/vbirding 发生错误时，才会使用 touch /tmp/vbirding 去建立这个档案的意思。 是否很有趣啊？这个 || 及 && 对于系统管理员在管理某些档案权限、存在等问题时， 可是很有用的东西喔！好了，现在我们来玩比较难一点的，看看底下的例题：

例题：以 ls 测试 /tmp/vbirding 是否存在，若存在则显示 "exist" ，若不存在，则显示 "not exist"！

答：

这又牵涉到逻辑判断的问题，如果存在就显示某个数据，若不存在就显示其它数据，那我可以这样做：

```
ls /tmp/vbirding && echo "exist" || echo "not exist"
```

意思是说，当 ls /tmp/vbirding 执行后，若正确，就执行 echo "exist" ，若有问题，就执行 echo "not exist" ！那如果我写成：

```
ls /tmp/vbirding || echo "not exist" && echo "exist"
```

对不对啊？这其实是有问题的，为什么呢？因为指令是一个一个往下执行，因此，在上面的例子当中，如果 /tmp/vbirding 不存在时，他会：

1. 若 ls /tmp/vbirding 不存在，因此回传一个非为 0 的数值；
2. 接下来经过 || 的判断，发现前一个指令回传非为 0 的数值，因此，程序开始执行 echo "not exist" ，而 echo "not exist" 程序肯定可以执行成功，因此会回传一个 0 值给后面的指令；
3. 经过 && 的判断，咦！是 0 啊！所以就开始执行 echo "exist" 。

所以啊，嘿嘿！第二个例子里面竟然会同时出现 not exist 与 exist 呢！真神奇～

经过这个范例的练习，您应该会了解，由于指令是一个接着一个去执行的，因此，如果真要使用判断， 那么这个 && 与 || 的顺序就不能搞错～一般来说，判断式最多会有三个，也就是：

```
command1 && command2 || command3
```

而且顺序通常不会变，因为一般来说， command2 与 command3 会放置肯定可以执行成功的指令， 因此，依据上面例题的逻辑分析，您就会晓得为何要如此放置啰～这很有用的啦！ 而且.....考试也很常考～

管线命令 (pipe)

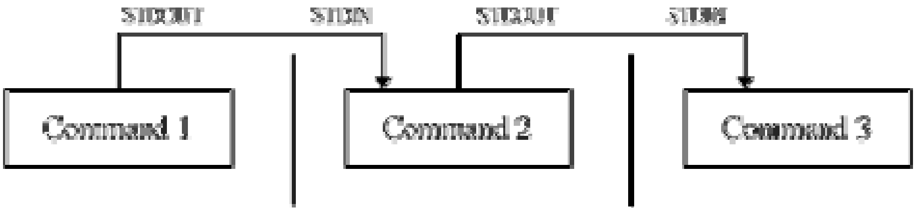
就如同前面所说的， bash 命令执行的时候有输出的数据会出现！ 那么如果这群数据必需要经过几道手续之后才能得到我们所想要的格式，应该如何来设定？ 这就牵涉到管线命令的问题了 (pipe) ，管线命令使用的是『 | 』这个界定符号！ 另外，管线命令与『连续下达命令』是不一样的呦！ 这点底下我们会再说明。底下我们先举一个例子来说明一下简单的管线命令。

假设我们想要知道 /etc/ 底下有多少档案，那么可以利用 ls /etc 来查阅，不过， 因为 /etc 底下的档案太多，导致一口气就将屏幕塞满了～不知道前面输出的内容是啥？此时，我们可以透过 less 指令的协助，利用：

```
[root@linux ~]# ls -al /etc | less
```

嘿嘿！如此一来，使用 ls 指令输出后的内容，就能够被 less 读取， 并且利用 less 的功能，我们就能够前后翻动相关的信息了！很方便是吧？呵呵！ 我们就来了解一下这个管线命令『 | 』的用途吧！

这个管线命令『 | 』仅能处理经由前面一个指令传来的正确信息，也就是 standard output (STDOUT) 的信息，对于 stdandard error 并没有直接处理的能力，请记得。那么整体的管线命令可以使用下图表示之：



图四、管线命令的处理示意图

在每个管线的前后部分都是『指令』呢！而后一个指令的输入乃是由前一个指令的输出而来的！ 不过，要注意的是，在 Linux 的环境中，很多的讯息处理都是以『行』为单位～ 也就是以是否具有 [Enter] 标志 (CR) 来作为一段处理的依据喔！ 底下我们来谈一谈一些基本的管线命令指令介绍：

撷取命令： cut, grep

什么是撷取命令啊？说穿了，就是将一段数据经过分析后，取出我们所想要的。 或者是，经由分析关键词，取得我们所想要的那一行！ 不过，要注意的是，一般来说，撷取讯息通常是针对『一行一行』来分析的，并不是整篇讯息分析的喔～底下我们介绍两个很常用的讯息撷取命令：

- cut

cut 不就是『切』吗？没错啦！这个指令可以将一段讯息的某一段给他『切』出来～ 处理的讯息是以『行』为单位喔！底下我们就来谈一谈：

```
[root@linux ~]# cut -d'分隔字符' -f fields
[root@linux ~]# cut -c 字符区间
参数：
-d  ：后面接分隔字符。与 -f 一起使用；
-f  ：依据 -d 的分隔字符将一段讯息分割成为数段，用 -f 取出第几段的意思；
-c  ：以字符 (characters) 的单位取出固定字符区间；
范例：

范例一：将 PATH 变量取出，我要找出第三个路径。
[root@linux ~]# echo $PATH
```

```
/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:/usr/X11R6/bin:/usr/games:
[root@linux ~]# echo $PATH | cut -d ':' -f 5
# 嘿嘿！如此一来，就会出现 /usr/local/bin 这个目录名称！
# 因为我们是以 : 作为分隔符，第五个就是 /usr/local/bin 啊！
# 那么如果想要列出第 3 与第 5 呢？，就是这样：
[root@linux ~]# echo $PATH | cut -d ':' -f 3,5

范例二：将 export 输出的讯息，取得第 12 字符以后的所有字符串
[root@linux ~]# export
declare -x HISTSIZE="1000"
declare -x INPUTRC="/etc/inputrc"
declare -x KDEDIR="/usr"
declare -x LANG="zh_TW.big5"
......其它省略......
[root@linux ~]# export | cut -c 12-
HISTSIZE="1000"
INPUTRC="/etc/inputrc"
KDEDIR="/usr"
LANG="zh_TW.big5"
......其它省略......
# 知道怎么回事了吧？用 -c 可以处理比较具有格式的输出数据！
# 我们还可以指定某个范围的值，例如第 12-20 的字符，就是 cut -c 12-20 等等！

范例三：用 last 将这个月登入者的信息中，仅留下使用者大名
[root@linux ~]# last
vbird  tty1  192.168.1.28   Mon Aug 15 11:55 - 17:48  (05:53)
vbird  tty1  192.168.1.28   Mon Aug 15 10:17 - 11:54  (01:37)
[root@linux ~]# last | cut -d ' ' -f 1
# 用 last 可以取得最近一个月登入主机的使用者信息，
# 而我们可以利用空格符的间隔，取出第一个信息，就是使用者账号啰！
# 但是因为 vbird tty1 之间空格有好几个，并非仅有一个，所以，如果要找出
# tty1 其实不能以 cut -d ' ' -f 1,2 喔！输出的结果会不是我们想要的。
```

这个 cut 实在很好用！不过，说真的，除非你常常在分析 log 档案，否则使用到 cut 的机会并不多！好了！ cut 主要的用途在于将『同一行里面的数据进行分解！』， 最常使用在分析一些数据或文字数据的时候！这是因为有时候我们会以某些字符当作分割的参数， 然后来将数据加以切割，以取得我们所需要的数据。我也很常使用这个功能呢！尤其是在分析 log 档案的时候！不过， cut 在处理多空格相连的数据时，可能会比较吃力一点～

- grep

刚刚的 cut 是将一行讯息当中，取出某部分我们想要的，而 grep 则是分析一行讯息， 若当中有我们所需要的信息，就将该行拿出来～简单的语法是这样的：

```
[root@linux ~]# grep [-acinv] '搜寻字符串' filename
```

```
参数：
-a ：将 binary 档案以 text 档案的方式搜寻数据
-c ：计算找到 '搜寻字符串' 的次数
-i ：忽略大小写的不同，所以大小写视为相同
-n ：顺便输出行号
-v ：反向选择，亦即显示出没有 '搜寻字符串' 内容的那一行！
范例：

范例一：将 last 当中，有出现 root 的那一行就取出来；
[root@linux ~]# last | grep 'root'

范例二：与范例一相反，只要没有 root 的就取出！
[root@linux ~]# last | grep -v 'root'

范例三：在 last 的输出讯息中，只要有 root 就取出，并且仅取第一栏
[root@linux ~]# last | grep 'root' |cut -d ' ' -f1
# 在取出 root 之后，利用上个指令 cut 的处理，就能够仅取得第一栏啰！
```

grep 是个很棒的指令喔！他支持的语法实在是太多了～用在正规表示法里头， 能够处理的数据实在是多的很～不过，我们这里先不谈正规表示法～下一章再来说明～ 您先了解一下， grep 可以解析一行文字，取得关键词，若该行有存在关键词， 就会整行列出来！

排序命令： sort, wc, uniq

很多时候，我们都会去计算一次数据里头的相同型态的数据总数，举例来说， 使用 last 可以查得这个月份有登入主机者的身份。那么我可以针对每个使用者查出他们的总登入次数吗？ 此时就得要排序与计算之类的指令来辅助了！底下我们介绍几个好用的排序与统计指令喔！

- sort

sort 是很有趣的指令，他可以帮我们进行排序，而且可以依据不同的数据型态来排序喔！ 例如数字与文字的排序就不一样。此外，排序的字符与语系的编码有关，因此， 如果您需要排序时，建议使用 LC_ALL=C 来让语系统一，数据排序比较好一些。

```
[root@linux ~]# sort [-fbMnrtuk] [file or stdin]
参数：
-f  ：忽略大小写的差异，例如 A 与 a 视为编码相同；
-b  ：忽略最前面的空格符部分；
-M  ：以月份的名字来排序，例如 JAN, DEC 等等的排序方法；
-n  ：使用『纯数字』进行排序(预设是以文字型态来排序的)；
-r  ：反向排序；
-u  ：就是 uniq ，相同的数据中，仅出现一行代表；
-t  ：分隔符，预设是 tab 键；
-k  ：以那个区间 (field) 来进行排序的意思，
范例：
```

```
范例一：个人账号都记录在 /etc/passwd 下，请将账号进行排序。
[root@linux ~]# cat /etc/passwd | sort
adm:x:3:4:adm:/var/adm:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
# 我省略很多的输出～由上面的数据看起来， sort 是预设『以第一个』数据来排序，
# 而且预设是以『文字』型态来排序的喔！所以由 a 开始排到最后啰！

范例二：/etc/passwd 内容是以 : 来分隔的，我想以第三栏来排序，该如何？
[root@linux ~]# cat /etc/passwd | sort -t ':' -k 3
root:x:0:0:root:/root:/bin/bash
iiimd:x:100:101:IIIMF server:/usr/lib/iiim:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
bin:x:1:1:bin:/bin:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
# 看到特殊字体的输出部分了吧？怎么会这样排列啊？呵呵！没错啦～
# 如果是以文字型态来排序的话，原本就会是这样，想要使用数字排序：
# cat /etc/passwd | sort -t ':' -k 3 -n
# 这样才行啊！用那个 -n 来告知 sort 以数字来排序啊！

范例三：利用 last ，将输出的数据仅取账号，并加以排序
[root@linux ~]# last | cut -d ' ' -f1 | sort
```

sort 同样是很常用的指令呢！因为我们常常需要比较一些信息啦！ 举个上面的第二个例子来说好了！今天假设你有很多的账号，而且你想要知道最大的使用者 ID 目前到哪一号了！呵呵！使用 sort 一下子就可以知道答案咯！当然其使用还不止此啦！ 有空的话不妨玩一玩！

- uniq

如果我排序完成了，想要将重复的资料仅列出一个显示，可以怎么做呢？

```
[root@linux ~]# uniq [-ic]
参数：
-i  ：忽略大小写字符的不同；
-c  ：进行计数
范例：

范例一：使用 last 将账号列出，仅取出账号栏，进行排序后仅取出一位；
[root@linux ~]# last | cut -d ' ' -f1 | sort | uniq

范例二：承上题，如果我还想要知道每个人的登入总次数呢？
[root@linux ~]# last | cut -d ' ' -f1 | sort | uniq -c
```

这个指令用来将『重复的行删除掉只显示一个』，举个例子来说， 你要知道这个月份登入你主机的使用者有谁，而不在乎他的登入次数，那么就使用上面的范例， (1)先将所有的数据列出；(2)再将人名独立出来；(3)经过排序；(4)只显示一个！ 由于这个指令是在将重复的东西减少，所以当然需要『配合排序过的档案』来处理啰！

- wc

如果我想要知道 /etc/man.config 这个档案里面有多少字？多少行？多少字符的话， 可以怎么做呢？其实可以利用 wc 这个指令来达成喔！他可以帮我们计算输出的讯息的整体数据！

```
[root@linux ~]# wc [-lwm]
参数：
-l  ：仅列出行；
-w  ：仅列出多少字(英文单字)；
-m  ：多少字符；
范例：

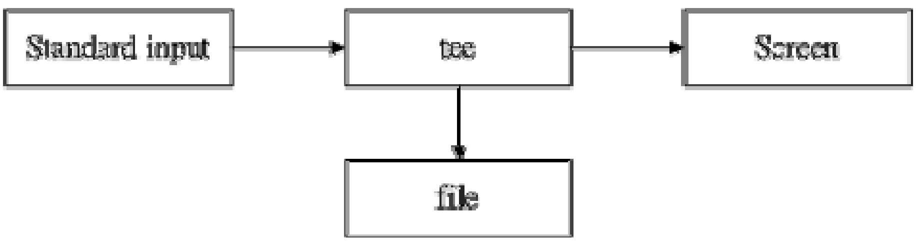
范例一：那个 /etc/man.config 里面到底有多少相关字、行、字符数？
[root@linux ~]# cat /etc/man.config | wc
    138     709    4506
# 输出的三个数字中，分别代表： 『行、字数、字符数』

范例二：我知道使用 last 可以输出登入者，但是 last 最后两行并非账号内容，
        那么请问，我该如何以一行指令串取得这个月份登入系统的总人次？
[root@linux ~]# last | grep [a-zA-Z] | grep -v 'wtmp' | wc -l
# 由于 last 会输出空白行与 wtmp 字样在最底下两行，因此，我利用
# grep 取出非空白行，以及去除 wtmp 那一行，在计算行数，就能够了解啰！
```

wc 也可以当作指令？呵呵！这可不是上洗手间的 WC 呢！ 这是相当有用的计算档案内容的一个工具组喔！举个例子来说， 当你要知道目前你的账号档案中有多少个账号时，就使用这个方法：『 cat /etc/passwd | wc -l 』啦！因为 /etc/passwd 里头一行代表一个使用者呀！ 所以知道行数就晓得有多少的账号在里头了！而如果要计算一个档案里头有多少个字符时，呵呵！就使用 wc -c 这个参数吧！

双向重导向： tee

想个简单的东西，我们由前一节知道 > 会将数据流整个传送给档案或装置， 因此我们除非去读取该档案或装置，否则就无法继续利用这个数据流。 万一我想要将这个数据流的处理过程中，将某段讯息存下来，应该怎么做？呵呵！ 利用 tee 就可以啰～我们可以这样简单的看一下：



图五、tee 的工作流程

同时将数据流分送到档案去与屏幕 (screen)；而输出到屏幕的，其实就是 stdout ，可以让下个指令继续处理喔！

```
[root@linux ~]# tee [-a] file
参数：
-a  ：以累加 (append) 的方式，将数据加入 file 当中！
范例：
[root@linux ~]# last | tee last.list | cut -d " " -f1
# 这个范例可以让我们将 last 的输出存一份到 last.list 档案中；
[root@linux ~]# ls -l /home | tee ~/homefile | more
# 这个范例则是将 ls 的数据存一份到 ~/homefile ，同时屏幕也有输出讯息！
[root@linux ~]# ls -l / | tee -a ~/homefile | more
# 要注意： tee 后接的档案会被覆盖，所以，我们要加上 -a
# 这个参数才能将讯息累加。
```

有没有发现在命令重导向的时候，如果我们要将数据送出到档案的时候， 屏幕上就不会出现任何的数据！那么如果我们需要将数据同时显示在屏幕上跟档案中呢？呵呵！这个时候就需要 tee 这个指令啰！使用 last 可以查看到这个月份的登入资料，而使用了 tee 之后，会将数据同时传给下一个命令去执行，也会将数据写入 last.list 这个档案中！也是个好帮手！

字符转换命令： tr, col, join, paste, expand

我们在 vi 文书处理器 章节当中，提到过 DOS 断行字符与 Unix 断行字符的不同， 并且可以使用 dos2unix 与 unix2dos 来完成转换。好了，那么思考一下，是否还有其它常用的字符替代？ 举例来说，要将大写改成小写，或者是 [tab] 按键转成空格键？还有，如何将两篇讯息整合成一篇？ 底下我们就来介绍一下这些字符转换命令在管线当中的使用方法：

- tr

tr 可以用来删除一段讯息当中的文字，或者是进行文字讯息的替换！

```
[root@linux ~]# tr [-ds] SET1 ...
参数：
-d  ：删除讯息当中的 SET1 这个字符串；
-s  ：取代掉重复的字符！
范例：

范例一：将 last 输出的讯息中，所有的小写变成大写字符：
[root@linux ~]# last | tr '[a-z]' '[A-Z]'

范例二：将 /etc/passwd 输出的讯息中，将冒号 (:) 删除
[root@linux ~]# cat /etc/passwd | tr -d ':'

范例三：将 DOS 档案的断行字符 ^M 符号删除：
```

```
[root@linux ~]# cat /home/test/dostxt | tr -d '\r' > dostxt-noM
# 那个 /r 指的是 DOS 的断行字符，关于更多的字符，请参考 man tr
```

其实这个指令也可以写在『正规表示法』里头！因为他也是由正规表示法的方式来取代数据的！ 以上面的例子来说，使用 [] 可以设定一串字呢！ 也常常用来取代档案中的怪异符号！ 例如上面第三个例子当中，可以去除 DOS 档案留下来的 ^M 这个断行的符号！这东西相当的有用！相信处理 Linux & Windows 系统中的人们最麻烦的一件事就是这个事情啦！亦即是 DOS 底下会自动的在每行行尾加入 ^M 这个断行符号！这个时候我们可以使用这个 tr 来将 ^M 去除！ ^M 可以使用 \r 来代替之！

- col

```
[root@linux ~]# col [-x]
参数：
-x  ：将 tab 键转换成对等的空格键
范例：
[root@linux ~]# cat -A /etc/man.config  <==此时会看到很多 ^I 的符号，那就是 tab
[root@linux ~]# cat /etc/man.config | col -x | cat -A | more
# 嘿嘿！如此一来， [tab] 按键会被取代成为空格键，输出就美观多了！
```

虽然 col 有他特殊的用途，不过，很多时候，他可以用来简单的处理将 [tab] 按键取代成为空格键！ 例如上面的例子当中，如果使用 cat -A 则 [tab] 会以 ^I 来表示。 但经过 col -x 的处理，则会将 [tab] 取代成为对等的空格键！

- join

join 看字面上的意义 (加入/参加) 就可以知道，他是在处理两个档案之间的数据， 而且，主要是在处理『两个档案当中，有 "相同数据" 的那一行，将他加在一起』的意思。我们利用底下的简单例子来说明：

```
[root@linux ~]# join [-ti12] file1 file2
参数：
-t  ：join 预设以空格符分隔数据，并且比对『第一个字段』的数据，
      如果两个档案相同，则将两笔数据联成一行，且第一个字段放在第一个！
-i  ：忽略大小写的差异；
-1  ：这个是数字的 1 ，代表『第一个档案要用那个字段来分析』的意思；
-2  ：代表『第二个档案要用那个字段来分析』的意思。
范例：

范例一：用 root 的身份，将 /etc/passwd 与 /etc/shadow 相关数据整合成一栏
[root@linux ~]# join -t ':' /etc/passwd /etc/shadow
bin:x:1:1:bin:/bin:/sbin/nologin:*:12959:0:99999:7:::
daemon:x:2:2:daemon:/sbin:/sbin/nologin:*:12959:0:99999:7:::
adm:x:3:4:adm:/var/adm:/sbin/nologin:*:12959:0:99999:7:::
# 因为 /etc/shadow 的权限问题，所以这里必须是 root 才能动作！而 /etc/passwd
# 与 /etc/shadow 都是以 : 来分隔字段，所以必须要使用 -t ':' 规范字段分隔字符。
# 且，因为 /etc/shadow 与 /etc/passwd 刚好都是以第一个字段为账号名称，所以，
```

```
# 就可以将同一行的数据给他贴在一起了！
# 另外，再仔细看一下 /etc/shadow 的内容与 /etc/passwd 的内容，您会发现，
# 两者都以账号为开始，而上面的输出数据中您会发现特殊字体部分，那代表
# 第二个档案的内容。在第二个档案的内容部分，由于账号(第一个字段)与
# 第一的档案是相同的，所以当然就省略掉，因此就成为上面的输出。

范例二：我们知道 /etc/passwd 第四个字段是 GID ，那个 GID 记录在
        /etc/group 当中的第三个字段，请问如何将两个档案整合？
[root@linux ~]# join -t ':' -1 4 /etc/passwd -2 3 /etc/group
0:root:x:0:root:/root:/bin/bash:root:x:
1:bin:x:1:bin:/bin:/sbin/nologin:bin:x:root,bin,daemon
2:daemon:x:2:daemon:/sbin:/sbin/nologin:daemon:x:root,bin,daemon
4:adm:x:3:adm:/var/adm:/sbin/nologin:adm:x:root,adm,daemon
# 这个例子就更明显了！原本的 /etc/passwd 的第一行内容应该是：
# root:x:0:0:root:/root:/bin/bash
# 至于 /etc/group 第一行内容应该是：
# root:x:0:
# 我将第一个档案的第四栏与第二个档案的第三栏取出，放置到输出的最前方，
# 然后将剩下的数据给他加在一起！就成了上面的输出啦！
```

这个 join 在处理两个相关的数据文件时，就真的是很有帮助的啦！ 例如上面的案例当中，我的 /etc/passwd, /etc/shadow, /etc/group 都是有相关性的， 其中 /etc/passwd, /etc/shadow 以账号为相关性，至于 /etc/passwd, /etc/group 则以所谓的 GID (账号的数字定义) 来作为他的相关性。根据这个相关性， 我们可以将有关系的资料放置在一起！这在处理数据可是相当有帮助的！ 但是上面的例子有点难，希望您可以静下心好好的看一看原因喔！

- paste

这个 paste 就要比 join 简单多了！相对于 join 必须要比对两个档案的数据相关性， paste 就直接『将两行贴在一起，且中间以 [tab] 键隔开』而已！简单的使用方法：

```
[root@linux ~]# paste [-d] file1 file2
参数：
-d  ：后面可以接分隔字符。预设是以 [tab] 来分隔的！
-   ：如果 file 部分写成 - ，表示来自 standard input 的资料的意思。
范例：

范例一：将 /etc/passwd 与 /etc/shadow 同一行贴在一起
[root@linux ~]# paste /etc/passwd /etc/shadow
bin:x:1:1:bin:/bin:/sbin/nologin        bin:*:12959:0:99999:7:::
daemon:x:2:2:daemon:/sbin:/sbin/nologin daemon:*:12959:0:99999:7:::
adm:x:3:4:adm:/var/adm:/sbin/nologin    adm:*:12959:0:99999:7:::
# 注意喔！同一行中间是以 [tab] 按键隔开的！

范例二：先将 /etc/group 读出(用 cat)，然后与范例一贴上一起！且仅取出前三行
```

```
[root@linux ~]# cat /etc/group|paste /etc/passwd /etc/shadow -|head -n 3
# 这个例子的重点在那个 - 的使用！那玩意儿常常代表 stdin 喔！
```

- expand

这玩意儿就是在将 [tab] 按键转成空格键啦～可以这样玩：

```
[root@linux ~]# expand [-t] file
参数：
-t  ：后面可以接数字。一般来说，一个 tab 按键可以用 8 个空格键取代。
      我们也可以自行定义一个 [tab] 按键代表多少个字符呢！
范例：

范例一：将 /etc/man.config 内行首为 MANPATH 的字样就取出；仅取前三行；
[root@linux ~]# grep '^MANPATH' /etc/man.config | head -n 3
MANPATH /usr/man
MANPATH /usr/share/man
MANPATH /usr/local/man
# 行首的代表标志为 ^ ，这个我们留待下节介绍！先有概念即可！

范例二：承上，如果我想要将所有的符号都列出来？(用 cat)
[root@linux ~]# grep '^MANPATH' /etc/man.config | head -n 3 |cat -A
MANPATH^I/usr/man$
MANPATH^I/usr/share/man$
MANPATH^I/usr/local/man$
# 发现差别了吗？没错～ [tab] 按键可以被 cat -A 显示成为 ^I

范例三：承上，我将 [tab] 按键设定成 6 个字符的话？
[root@linux ~]# grep '^MANPATH' /etc/man.config | head -n 3 | \
> expand -t 6 - | cat -A
MANPATH     /usr/man$
MANPATH     /usr/share/man$
MANPATH     /usr/local/man$
123456123456123456.....
# 仔细看一下上面的数字说明，因为我是以 6 个字符来代表一个 [tab] 的长度，所以，
# MAN... 到 /usr 之间会隔 12 (两个 [tab]) 个字符喔！如果 tab 改成 9 的话，
# 情况就又不同了！这里也不好理解～您可以多设定几个数字来查阅就晓得！
```

expand 也是挺好玩的～他会自动将 [tab] 转成空格键～所以，以上面的例子来说， 使用 cat -A 就会查不到 ^I 的字符啰～此外，因为 [tab] 最大的功能就是格式排列整齐！ 我们转成空格键后，这个空格键也会依据我们自己的定义来增加大小～ 所以，并不是一个 ^I 就会换成 8 个空白喔！这个地方要特别注意的哩！ 此外，您也可以参考一下 unexpand 这个将空白转成 [tab] 的指令功能啊！ ^_^

分割命令： split

如果你有档案太大，导致一些携带式装置无法复制的问题，嘿嘿！找 split 就对了！ 他可以帮你将一个大档案，依据档案大小或行数来分割，就可以将大档案分割成为小档案了！ 快速又有效啊！真不错～

```
[root@linux ~]# split [-bl] file PREFIX
参数：
-b  ：后面可接欲分割成的档案大小，可加单位，例如 b, k, m 等；
-l  ：以行数来进行分割。
范例：

范例一：我的 /etc/termcap 有七百多 K，若想要分成 300K 一个档案时？
[root@linux ~]# cd /tmp; split -b 300k /etc/termcap termcap
[root@linux tmp]# ls -l termcap*
-rw-rw-r--  1 root root  307200  8 月 17 00:25 termcapaa
-rw-rw-r--  1 root root  307200  8 月 17 00:25 termcapab
-rw-rw-r--  1 root root  184848  8 月 17 00:25 termcapac
# 那个档名可以随意取的啦！我们只要写上前导文字，小档案就会以
# xxxaa, xxxab, xxxac 等方式来建立小档案的！

范例二：如何将上面的三个小档案合成一个档案，档名为 termcapback
[root@linux tmp]# cat termcap* >> termcapback
# 很简单吧？就用数据流重导向就好啦！简单！

范例三：使用 ls -al / 输出的信息中，每十行记录成一个档案
[root@linux tmp]# ls -al / | split -l 10 - lsroot
# 重点在那个 - 啦！一般来说，如果需要 stdout/stdin 时，但偏偏又没有档案，
# 有的只是 - 时，那么那个 - 就会被当成 stdin 或 stdout ～
```

在 Windows 的情况下，你要将档案分割需要如何作？！伤脑筋吧！呵呵！在 Linux 底下就简单的多了！你要将档案分割的话，那么就使用 -b size 来将一个分割的档案限制其大小，如果是行数的话，那么就使用 -l line 来分割！好用的很！如此一来，你就可以轻易的将你的档案分割成 floppy 的大小，方便你 copy 啰！

参数代换： xargs

xargs 是在做什么的呢？就以字面上的意义来看， x 是加减乘除的乘号，args 则是 arguments (参数) 的意思，所以说，这个玩意儿就是在产生某个指令的参数的意思！ xargs 可以读入 stdin 的数据，并且以空格符或断行字符作为分辨，将 stdin 的资料分隔成为 arguments 。 因为是以空格符作为分隔，所以，如果有一些档名或者是其它意义的名词内含有空格符的时候， xargs 可能就会误判了～他的用法其实也还满简单的！就来看一看先！

```
[root@linux ~]# xargs [-0epn] command
参数：
-0  ：如果输入的 stdin 含有特殊字符，例如 `, \, 空格键等等字符时，这个 -0 参数
      可以将他还原成一般字符。这个参数可以用于特殊状态喔！
-e  ：这个是 EOF (end of file) 的意思。后面可以接一个字符串，当 xargs 分析到
```

```
      这个字符串时，就会停止继续工作！
-p  ：在执行每个指令的 argument 时，都会询问使用者的意思；
-n  ：后面接次数，每次 command 指令执行时，要使用几个参数的意思。看范例三。
当 xargs 后面没有接任何的指令时，预设是以 echo 来进行输出喔！
范例：

范例一：将 /etc/passwd 内的第一栏取出，仅取三行，使用 finger 这个指令将每个
        账号内容秀出来
[root@linux ~]# cut -d':' -f1 < /etc/passwd |head -n 3| xargs finger
Login: root                             Name: root
Directory: /root                        Shell: /bin/bash
Never logged in.
No mail.
No Plan.
......底下省略.....
# 由 finger account 可以取得该账号的相关说明内容，例如上面的输出就是 finger root
# 后的结果。在这个例子当中，我们利用 cut 取出账号名称，用 head 取出三个账号，
# 最后则是由 xargs 将三个账号的名称变成 finger 后面需要的参数！

范例二：同上，但是每次执行 finger 时，都要询问使用者是否动作？
[root@linux ~]# cut -d':' -f1 < /etc/passwd |head -n 3| xargs -p finger
finger root bin daemon ?...y
......底下省略.....
# 呵呵！这个 -p 的参数有趣了吧？！他可以让使用者的使用过程中，被询问到每个
# 指令是否执行！

范例三：将所有的 /etc/passwd 内的账号都以 finger 查阅，但一次仅查阅五个账号
[root@linux ~]# cut -d':' -f1 < /etc/passwd | xargs -p -n 5 finger
finger root bin daemon adm lp ?...y
......底下省略.....
# 在这里鸟哥使用了 -p 这个参数来让您对于 -n 更有概念。一般来说，某些指令后面
# 可以接的 arguments 是有限制的，不能无限制的累加，此时，我们可以利用 -n
# 来帮助我们将参数分成数个部分，每个部分分别再以指令来执行！这样就 OK 啦！^_^
[root@linux ~]#

范例四：同上，但是当分析到 lp 就结束这串指令？
[root@linux ~]# cut -d':' -f1 < /etc/passwd | xargs -p -e'lp' finger
finger root bin daemon adm ?...
# 仔细与上面的案例做比较。也同时注意，那个 -e'lp' 是连在一起的，中间没有空格键。
# 上个例子当中，第五个参数是 lp 啊，那么我们下达 -e'lp' 后，则分析到 lp
# 这个字符串时，后面的其它 stdin 的内容就会被 xargs 舍弃掉了！
```

其实，在 man xargs 里面就有三四个小范例，您可以自行参考一下内容。 此外， xargs 真的是很好用的一个玩意儿！您真的需要好好的参详参详！

关于减号 - 的用途

管线命令在 bash 的连续的处理程序中是相当重要的！另外，在 log file 的分析当中也是相当重要的一环， 所以请特别留意！另外，在管线命令当中，常常会使用到前一个指令的 stdout 作为这次的 stdin ，某些指令需要用到文件名称 (例如 tar) 来进行处理时，该 stdin 与 stdout 可以利用减号 "-" 来替代，举例来说：

```
[root@linux ~]# tar -cvf - /home | tar -xvf -
```

上面这个例子是说：『我将 /home 里面的档案给他打包，但打包的数据不是纪录到档案，而是传送到 stdout； 经过管线后，将 tar -cvf - /home 传送给后面的 tar -xvf - 』。后面的这个 - 则是取用前一个指令的 stdout， 因此，我们就不需要使用 file 了！这是很常见的例子喔！注意注意！

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 在 Linux 上可以找到哪些 shell(举出三个) ？那个档案记录可用的 shell ？而 Linux 预设的 shell 是？

  1) /bin/bash, /bin/tcsh, /bin/csh
  2) /etc/shells
  3) bash ，亦即是 /bin/bash。

- 在 shell 环境下，有个提示字符 (prompt)，他可以修改吗？要改什么？预设的提示字符内容是？

  可以修改的，改 PS1 这个变量，这个 PS1 变量的预设内容为：『[\u@\h \W]\$』

- 如何显示 HOME 这个环境变量？

  echo $HOME

- 如何得知目前的所有变量与环境变量的设定值？

  环境变量用 env 而所有变量用 set 即可显示

- 我是否可以设定一个变量名称为 3myhome ？

  不行！变量不能以数字做为开头，参考变量设定规则的内容

- 在这样的练习中『A=B』且『B=C』，若我下达『unset $A』，则取消的变数是 A 还是 B?

  被取消的是 B 喔，因为 unset $A 相当于 unset B 所以取消的是 B ，A 会继续存在！

- 如何取消变量与命令别名的内容？

使用 unset 及 unalias 即可

- 如何设定一个变量名称为 name 内容为 It's my name ？

name=It\'s\ my\ name 或 name="It's my name"

- 环境变量档案的加载顺序？

先由 /etc/passwd 取得 bash 这个 shell ，再到 /etc/profile 读取主要的环境变量，同时亦会将 /etc/inputrc 及 /etc/profile.d 内容均读入。之后，再到个人的家目录读取 ~/.bash_profile 及 ~/.bashrc 等档案！

- man page 的路径设定档案？

/etc/man.config 或 /etc/man.conf

- 试说明 ', ", 与 ` 这些符号在变量定义中的用途？

参考变量规则那一章节，其中， " 可以具有变量的内容属性， ' 则仅有一般字符，至于 ` 之内则是可先被执行的指令。

- 跳脱符号 \ 有什么用途？

可以用来跳脱特殊字符，例如 Enter, $ 等等，使成为一般字符！

- 连续命令中， ;, &&, || 有何不同？

分号可以让两个 command 连续运作，不考虑 command1 的输出状态， && 则前一个指令必需要没有错误讯息，亦即回传值需为 0 则 command2 才会被执行， || 则与 && 相反！

- 如何将 last 的结果中，独立出账号，并且印出本月份曾经登入过的账号？

last | cut -d " " -f1 | sort | uniq

- 请问 foo1 && foo2 | foo3 > foo4 ，这个指令串当中， foo1/foo2/foo3/foo4 是指令还是档案？整串指令的意义为？

foo1/foo2 与 foo3 都是指令， foo4 是装置或档案。整串指令意义为：

1. 当 foo1 执行结果有错误时，则该指令串结束；
2. 若 foo1 执行结果没有错误时，则执行 foo2 | foo3 > foo4 ；
    1. foo2 将 stdout 输出的结果传给 foo3 处理；
    2. foo3 将来自 foo2 的 stdout 当成 stdin ，处理完后将数据流重新导向 foo4 这个装置/档案

- 如何秀出在 /bin 底下任何以 a 为开头的档案文件名的详细资料？

```
ls -l /bin/a*
```

- 如何秀出 /bin 底下，文件名为四个字符的档案？

  ```
  ls -l /bin/????
  ```

- 如何秀出 /bin 底下，档名开头不是 a-d 的档案？

  ```
  ls -l /bin/[!a-d]*
  ```

- 当我离开 bash 后，希望系统可以帮我将最近工作的：1.)工作日期； 2.)100 个历史命令独立 记录到 ~/.bash_localcom 档案中，该如何设定？

  我可以编辑 ~/.bash_logout ，将这个档案内容变成：

  ```
  # ~/.bash_logout
  date >> ~/.bash_localcom
  history 100 >> ~/.bash_localcom
  clear
  ```

- 我想要让终端机接口的登入提示字符修改成我自己喜好的模样，应该要改哪里？(filename)

  /etc/issue

- 承上题，如果我是想要让使用者登入后，才显示欢迎讯息，又应该要改哪里？

  /etc/motd

参考数据

- 卧龙小三的教学文件：http://linux.tnc.edu.tw/techdoc/shell/book1.html
- GNU 计划的 BASH 说明：http://www.gnu.org/manual/bash-2.05a/html_mono/bashref.html
  鸟哥的备份：http://linux.vbird.org/linux_basic/0320bash/0320bash_reference.php
- man bash

正规表示法 Regular Expression, RE

最近更新日期：2005/09/05

正规表示法(或称为常规表示法)是透过一些特殊字符的排列，用以 搜寻/取代/删除 一列或多列文字字符串， 简单的说，正规表示法就是用在字符串的处理上面的一项『表示式』。正规表示法并不是一个工具程序， 而是一个字符串处理的标准依据，如果您想要以正规表示法的方式处理字符串，就得要使用支持正规表示法的工具程序才行， 这类的工具程序很多，例如 vi, sed, awk 等等。

正规表示法对于系统管理员来说，实在是很重要。因为系统会产生很多的讯息，这些讯息有的重要，有的仅是告知， 此时，管理员可以透过正规表示法的功能来将重要讯息撷取出来，并产生便于查阅的报表，简化管理流程。此外， 很多的软件包也都支持正规表示法的分析，例如邮件服务器的过滤机制(过滤垃圾信件)就是很重要的一个例子。 所以，您最好要了解正规表示法的相关技能，在未来管理主机时，才能够更精简处理您的日常事务！

注：本章节使用者需要多加练习，因为目前很多的套件都是使用正规表示法来达成其『过滤、分析』的目的， 为了未来主机管理的便利性，使用者至少要能看的懂正规表示法的意义！

前言

约略了解了 Linux 的基本指令 (Shell) 并且熟悉了 vi 之后，相信您对于敲击键盘与指令比较不陌生了吧？？ 接下来，底下要开始介绍一个很重要的观念，那就是所谓的『正规表示法』啰！

什么是正规表示法

任何一个有经验的系统管理员，都会告诉您：『正规表示法真是挺重要的！』 为什么很重要呢？因为日常生活就使用的到啊！举个例子来说， 在您日常处理文书作业时，应该会常常使用到『搜寻/取代』等等的功能吧？ 这些举动要作的漂亮，就是正规表示法的工作了！

简单的说，正规表示法就是处理字符串的方法，他是以行为单位， 来进行字符串的处理行为，他透过一些特殊符号的辅助，可以让使用者轻易的达到 搜寻/取代 某特定字符串的处理程序！

举例来说，我要找到 VBird 或 Vbird 这个字样，但是不要其它的字符串，该如何办理？ 如果在没有正规表示法的环境中(例如 MS word)，您或许就得要使用忽略大小写的办法， 或者是分别以 VBird 及 Vbird 搜寻两遍。但是，忽略大小写可能会搜寻到 VBIRD/vbird/VbIrD 等等的不需要的字符串，而造成使用者的困扰。

再举个系统常见的例子好了，假设妳发现系统在开机的时候，老是会出现一个关于 mail 程序的错误， 而开机过程的相关程序都是在 /etc/rc.d/ 底下，也就是说，在该目录底下的某个档案内具有 mail 这个关键词，好了，此时，您怎么找出来含有这个关键词的档案？？您当然可以一个档案一个档案的开启， 然后去搜寻 mail 这个关键词，只是.....该目录底下的档案可能不止 100 个说～ 如果了解正规表示法的相关技巧，那么只要一行指令就找出来啦！ 『grep 'mail' /etc/rc.d/*』 那个 grep 就是支持正规表示法的工具程序之一！如何～很简单吧！ ^_^y

谈到这里就得要进一步说明了，正规表示法基本上是一种『表示法』， 只要工具程序支持这种表示法，那么该工具程序就可以用来作为正规表示法的字符串处理之用。也就是说，例如 vi, grep, awk ,sed 等等工具，因为她们有支持正规表示法， 所以，这些工具就可以使用正规表示法的特殊字符来进行字符串的处理。

正规表示法对于系统管理员的用途

那么为何我需要学习正规表示法呢？对于一般使用者来说，由于使用到正规表示法的机会可能不怎么多，因此感受不到他的魅力，不过，对于身为系统管理员的您来说， 正规表示法则是一个『不可不学的好东西！』 怎么说呢？由于系统如果在繁忙的情况之下，每天产生的讯息信息会多到你无法想象的地步， 而我们也都知道，系统的『错误讯息登录档案』 的内容(这部份我们在第五篇会详谈)记载了系统产生的所有讯息，当然， 这包含你的系统是否被『入侵』的纪录数据。

但是系统的数据量太大了，要身为系统管理员的你每天去看这么多的讯息数据， 从千百行的资料里面找出一行有问题的讯息，呵呵～光是用肉眼去看，想不疯掉都很难！ 这个时候，我们就可以透过『正规表示法』的功能，将这些登录的信息进行处理， 仅取出『有问题』的信息来进行分析，哈哈！如此一来，你的系统管理工作将会 『快乐得不得了』啊！当然，正规表示法的优点还不止于此，等您有一定程度的了解之后，您会爱上他喔！

正规表示法的广泛用途

正规表示法除了可以让系统管理员管理主机更为便利之外，事实上， 由于正规表示法强大的字符串处理能力，目前一堆软件都支持正规表示法呢！ 最常见的就是『邮件服务器』啦！

如果您留意因特网上的消息，那么应该不能发现，目前造成网络大塞车的主因之一就是『垃圾/广告信件』了， 而如果我们可以在主机端，就将这些问题邮件剔除的话，客户端就会减少很多不必要的频宽耗损了。 那么如何剔除广告信件呢？由于广告信件几乎都有一定的标题或者是内容，因此， 只要每次有来信时，都先将来信的标题与内容进行特殊字符串的比对，发现有不良信件就予以剔除！ 嘿！这个工作怎么达到啊？就使用正规表示法啊！目前两大邮件服务器软件 sendmail 与 postfix 以及支持邮件服务器的相关分析套

件，都支持正规表示法的比对功能！

当然还不止于此啦，很多的服务器软件、以及套件都支持正规表示法呢！当然， 虽然各家软件都支持他，不过，这些『字符串』的比对还是需要系统管理员来加入比对规则的， 所以啦！身为系统管理员的你，为了自身的工作以及客户端的需求， 正规表示法实在是很需要也很值得学习的一项工具呢！

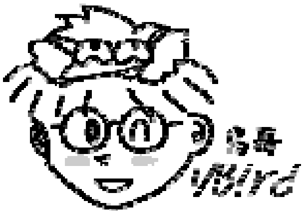
正规表示法与 Shell 在 Linux 当中的角色定位

说实在的，我们在学数学的时候，一个很重要、但是粉难的东西是一定要『背』的， 那就是九九表，背成功了之后，未来在数学应用的路途上，真是一帆风顺啊！ 这个九九表我们在小学的时候几乎背了一整年才背下来，并不是这么好背的呢！ 但他却是基础当中的基础！您现在一定受惠相当的多呢 ^_^！ 而我们谈到的这个正规表示法，与前一章的 BASH shell 就有点像是数学的九九表一样，是 Linux 基础当中的基础，虽然也是最难的部分， 不过，如果学成了之后，一定是『大大的有帮助』的！这就好像是金庸小说里面的学武难关， 任督二脉，打通任督二脉之后，武功立刻成倍成长！所以啦， 不论是对于系统的认识与系统的管理部分，他都有很棒的辅助啊！请好好的学习这个基础吧！ ^_^

延伸的正规表示法

正规表示法除了简单的一组字符串处理之外，还可以作群组的字符串处理， 例如进行搜寻 VBird 或 netman 或 lman 的搜寻，注意，是『或(or)』而不是『和(and)』的处理， 此时就需要延伸正规表示法的帮助啦！藉由特殊的 (与 | 等字符的协助， 就能够达到这样的目的！好啦！清清脑门，咱们用功去啰！

> Tips:
> 有一点要向大家报告的，那就是：『正规表示法与万用字符是不一样的东西！』 这很重要喔！因为万用字符 (wildcard) 所代表的意义与正规表示法并不相同～ 要分的很清楚才行喔！所以，学习本章，请将前一章 bash 的万用字符意义先忘掉吧！



基础正规表示法

既然正规表示法是处理字符串的一个标准表示方式，他需要支持的工具程序来辅助， 所以，我们这里就先介绍一个最简单的字符串撷取功能的工具程序，那就是 grep 啰！ 在介绍完 grep 的基本功能之后，就进入正规表示法的特殊字符的处理能力了。

以 grep 撷取字符串

既然要使用 grep 当然就得要先了解一下 grep 的语法啰～

```
[root@test root]# grep [-acinv] '搜寻字符串' filename
参数说明：
-a ：将 binary 档案以 text 档案的方式搜寻数据
-c ：计算找到 '搜寻字符串' 的次数
-i ：忽略大小写的不同，所以大小写视为相同
-n ：顺便输出行号
```

```
-v ：反向选择，亦即显示出没有 '搜寻字符串' 内容的那一行！
范例：
[root@test root]# grep 'root' /var/log/secure
将 /var/log/secure 这个档案中有 root 的那一行秀出来

[root@test root]# grep -v 'root' /var/log/secure
若该行没有 root 才将数据秀出来到屏幕上！

[root@test root]# last | grep root
若该行有 root 才将数据秀出来到屏幕上！
```

grep 是一个很常见也很常用的指令，他最重要的功能就是进行字符串数据的比对， 然后将符合使用者需求的字符串打印出来。 需要说明的是『grep 在数据中查寻一个字符串时，是以 "整行" 为单位来进行数据的撷取的！』也就是说，假如一个档案内有 10 行，其中有两行具有你所搜寻的字符串，则将那两行显示在屏幕上，其它的就丢弃了！

而 grep 除了可以进行档案的资料搜寻之外，也常常被应用在 input/output 的数据处理当中，例如常见的管线命令 (pipe) 就可以常常见到他的踪影！ 以上面表格中的例子来看，我们可以发现前两个例子是查寻档案的内容，有没有加上 -v 所显示出来的结果是『相反的！』，而第三个例子则是以 pipe 的功能进行数据的处理的喔！

好了，我们就开始以 grep 来进行正规表示法的简易说明吧！我们先以底下这个档案来作为范例：

```
[root@test root]# vi regular_express.txt
"Open Source" is a good mechanism to develop programs.
apple is my favorite food.
Football game is not use feet only.
this dress doesn't fit me.
However, this dress is about $ 3183 dollars.
GNU is free air not free beer.
Her hair is very beauty.
I can’t finish the test.
Oh! The soup taste good.
motorcycle is cheap than car.
This window is clear.
the symbol '*' is represented as start.
Oh! My god!
The gd software is a library for drafting programs.
You are the best is mean you are the no. 1.
The world is the same with "glad".
I like dog.
google is the best tools for search keyword.
goooooogle yes!
go! go! Let's go.
```

```
# I am VBird
```

需要特别注意的是，上面这个档案鸟哥是在 Windows 的环境下编辑的， 并且经过特殊处理过，因此，他虽然是纯文字文件，但是内含一些 Windows 环境下的软件常常自行加入的一些特殊字符，例如断行字符 (^M)就是一例！ 所以，您可以直接将上面的文字以 vi 储存成 regular_express.txt 这个档案， 不过，比较建议直接点底下的连结下载：

http://linux.vbird.org/linux_basic/0330regularex/regular_express.txt

此外，因为不同的语系编码是不一样的，所以，您必须要将语系改成英文语系， 才能够进行底下的测试，否则，可能会有显示的内容与底下的输出不符的状况喔！ 修改语系的方法为：

```
[root@test root]# LANG=en
[root@test root]# export LANG
```

好了，现在开始我们一个案例一个案例的来介绍吧！

- 例题一、搜寻特定字符串：
  搜寻特定字符串很简单吧？假设我们要从刚刚的档案当中取得 the 这个特定字符串， 最简单的方式就是这样：

```
[root@test root]# grep -n 'the' regular_express.txt
8:I can't finish the test.
12:the symbol '*' is represented as start.
15:You are the best is mean you are the no. 1.
16:The world  is the same with "glad".
18:google is the best tools for search keyword.
```

那如果想要『反向选择』呢？也就是说，当该行没有 'the' 这个字符串时，才显示在屏幕上，那就直接使用：

```
[root@test root]# grep -vn 'the' regular_express.txt
```

您会发现，屏幕上出现的行列为除了 8,12,15,16,18 五行之外的其它行列！ 接下来，如果您想要取得不论大小写的 the 这个字符串，则：

```
[root@test root]# grep -in 'the' regular_express.txt
8:I can't finish the test.
9:Oh! The soup taste good.
12:the symbol '*' is represented as start.
14:The gd software is a library for drafting programs.
15:You are the best is mean you are the no. 1.
16:The world  is the same with "glad".
18:google is the best tools for search keyword.
```

- 例题二、利用 [] 来搜寻集合字符

  如果我想要搜寻 test 或 taste 这两个单字时，可以发现到，其实她们有共通的 't?st' 存在～这个时候，我可以这样来搜寻：

```
[root@test root]# grep -n 't[ae]st' regular_express.txt
8:I can't finish the test.
9:Oh! The soup taste good.
```

了解了吧？其实 [] 里面不论有几个字符，他都谨代表某『一个』字符， 所以，上面的例子说明了，我需要的字符串是『tast』或『test』两个字符串而已！ 而如果想要搜寻到有 oo 的字符时，则使用：

```
[root@test root]# grep -n 'oo' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
9:Oh! The soup taste good.
18:google is the best tools for search keyword.
19:goooooogle yes!
```

但是，如果我不想要 oo 前面有 g 的话呢？此时，可以利用在集合字符的反向选择 [^] 来达成

```
[root@test root]# grep -n '[^g]oo' regular_express.txt
2:apple is my favorite food.
3:Football game is not use feet only.
18:google is the best tools for search keyword.
19:goooooogle yes!
```

意思就是说，我需要的是 oo ，但是 oo 前面不能是 g 就是了！ 仔细比较上面两个表格，妳会发现，第 1,9 行不见了，因为 oo 前面出现了 g 所致！ 第 2,3 行没有疑问，因为 foo 与 Foo 均可被接受！但是第 18 行明明有 google 的 goo 啊～ 别忘记了，因为该行后面出现了 tool 的 too 啊！ 所以该行也被列出来～ 也就是说， 18 行里面虽然出现了我们所不要的项目 (goo) 但是由于有需要的项目 (too) ， 因此，是符合字符串搜寻的喔！

至于第 19 行，同样的，因为 goooooogle 里面的 oo 前面可能是 o ，例如： go(ooo)oogle ，所以，这一行也是符合需求的！

再来，假设我 oo 前面不想要有小写字符，所以，我可以这样写 [^abcd....z]oo ， 但是这样似乎不怎么方便，由于小写字符的 ASCII 上编码的顺序是连续的， 因此，我们可以将之简化为底下这样：

```
[root@test root]# grep -n '[^a-z]oo' regular_express.txt
3:Football game is not use feet only.
```

也就是说，当我们在一组集合字符中，如果该字符组是连续的，例如大写英文/小写英文/数字等等，就可以使用[a-z],[A-Z],[0-9]等方式来书写，那么如果我们的要求字符串是数字与英文呢？ 呵呵！就将他全部写在一起，变成：[a-zA-Z0-9]

例如，我们要取得有数字的那一行，就这样：

```
[root@test root]# grep -n '[0-9]' regular_express.txt
5:However, this dress is about $ 3183 dollars.
15:You are the best is mean you are the no. 1.
```

这样对于 [] 以及 [^] 以及 [] 当中的 - 有了解了吗？！ ^_^y

- 例题三、行首与行尾字符 ^ $：
我们在例题一当中，可以查询到一行字符串里面有 the 的，那如果我想要让 the 只在行首列出呢？ 这个时候就得要使用定位字符了！我们可以这样做：

```
[root@test root]# grep -n '^the' regular_express.txt
12:the symbol '*' is represented as start.
```

此时，就只剩下第 12 行，因为只有第 12 行的行首是 the 开头啊～此外， 如果我想要开头是小写字符的那一行就列出呢？可以这样：

```
[root@test root]# grep -n '^[a-z]' regular_express.txt
2:apple is my favorite food.
4:this dress doesn't fit me.
10:motorcycle is cheap than car.
12:the symbol '*' is represented as start.
18:google is the best tools for search keyword.
19:goooooogle yes!
```

如果我不想要开头是英文字母，则可以是这样：

```
[root@test root]# grep -n '^[^a-zA-Z]' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
20:# I am VBird
```

注意到了吧？那个 ^ 符号，在字符集合符号(括号[])之内与之外是不同的！ 在 [] 内代表『反向选择』，在 [] 之外则代表定位在行首的意义！要分清楚喔！

那如果我想要找出来，行尾结束为小数点 (.) 的那一行，该如何处理：

```
[root@test root]# grep -n '\.$' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
4:this dress doesn't fit me.
10:motorcycle is cheap than car.
11:This window is clear.
12:the symbol '*' is represented as start.
15:You are the best is mean you are the no. 1.
16:The world  is the same with "glad".
17:I like dog.
18:google is the best tools for search keyword.
```

特别注意到，因为小数点具有其它意义(底下会介绍)，所以必须要使用跳脱字符(\)来加以解除其特殊意义！ 不过，您或许会觉得奇怪，但是第 5~9 行最后面也是 . 啊～怎么无法打印出来？？ 这里就牵涉到 Windows 平台的软件对于断行字符的判断问题了！我们使用 cat -A 将第五行拿出来看， 您会发现：

```
[root@test root]# cat -A regular_express.txt
However, this dress is about $ 3183 dollars.^M$
```

注意到了没？最后面的断行字符应该是 $ 才对，但是，因为 Windows 的 nodepad 会主动加上 ^M 作为断行的判断，因此，那个 . 自然就不是紧接在 $ 之前喔！这样可以了解 ^ 与 $ 的意义吗？ 好了，先不要看底下的解答，自己想一想，那么如果我想要找出来，哪一行是『空白行』， 也就是说，该行并没有输入任何数据，该如何搜寻？？

```
[root@test root]# grep -n '^$' regular_express.txt
21:
```

因为只有行首跟行尾(^$)，所以，这样就可以找出空白行啦！再来， 假设您已经知道在一个批次脚本 (shell script) 或者是设定档当中， 空白行与开头为 # 的那一行是批注，因此如果您要将资料列出给别人参考时， 可以将这些数据省略掉，以节省保贵的纸张，那么，您可以怎么作呢？ 我们以 /etc/syslog.conf 这个档案来作范例，您可以自行参考一下输出的结果：

```
[root@test root]# cat /etc/syslog.conf
[root@test root]# grep -v '^$' /etc/syslog.conf | grep -v '^#'
```

是否节省很多版面啊？？

- 例题四、任意一个字符 . 与重复字符 *

在 bash 的章节当中，我们知道万用字符 * 可以用来代表任意(0 或多个)字符， 但是正规表示法并不是万用字符，两者之间是不相同的！ 至于正规表示法当中的『 . 』则代表『绝对有一个任意字符』的意思！这样讲不好懂， 我们直接做个练习吧！假设我需要找出 g??d 的字符串，亦即共有四个字符， 起头是 g 而结束是 d ，我可以这样做：

```
[root@test root]# grep -n 'g..d' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
9:Oh! The soup taste good.
16:The world  is the same with "glad".
```

因为强调 g 与 d 之间一定要存在两个字符，因此，第 13 行的 god 与第 14 行的 gd 就不会被列出来啦！再来，如果我想要列出有 oo, ooo, oooo 等等的数据， 也就是说，至少要有两个 o 以上，该如何是好？？是 o* 还是 oo* 还是 ooo* 呢？ 虽然您可以试看看结果， 不过结果太占版面了 @_@ ，所以，我这里就直接说明。

因为 * 代表的是『重复 0 个或多个前面的 RE 字符』的意义， 因此，『o*』代表的是：『拥有空字符或一个 o 以上的字符』， 特别注意，因为允许空字符(就是有没有字符都可以的意思)，因此， grep -n 'o*' regular_express.txt 将会把所有的数据都打印出来屏幕上！

那如果是『oo*』呢？则第一个 o 肯定必须要存在，第二个 o 则是可有可无的多个 o ， 所以，凡是含有 o, oo, ooo, oooo 等等，都可以被列出来～

同理，当我们需要『至少两个 o 以上的字符串』时，就需要 ooo* ，亦即是：

```
[root@test root]# grep -n 'ooo*' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
9:Oh! The soup taste good.
18:google is the best tools for search keyword.
19:goooooogle yes!
```

这样理解 * 的意义了吗？！好了，现在出个练习，如果我想要字符串开头与结尾都是 g， 但是两个 g 之间仅能存在至少一个 o ，亦即是 gog, goog, gooog.... 等等， 那该如何？

```
[root@test root]# grep -n 'goo*g' regular_express.txt
18:google is the best tools for search keyword.
19:goooooogle yes!
```

如此了解了吗？好，再来一题，如果我想要找出 g 开头与 g 结尾的字符串， 当中的字符可有可无，那该如何是好？是『g*g』吗？

```
[root@test root]# grep -n 'g*g' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
3:Football game is not use feet only.
9:Oh! The soup taste good.
13:Oh!  My god!
14:The gd software is a library for drafting programs.
16:The world  is the same with "glad".
17:I like dog.
18:google is the best tools for search keyword.
19:goooooogle yes!
```

但测试的结果竟然出现这么多行？？太诡异了吧？ 其实一点也不诡异，因为 g*g 里面的 g* 代表『空字符或一个以上的 g』 在加上后面的 g ，因此，整个 RE 的内容就是 g, gg, ggg, gggg ，因此，只要该行当中拥有一个以上的 g 就符合所需了！

那该如何得到我们的 g....g 的需求呢？呵呵！就利用任意一个字符『.』啊！ 亦即是：『g.*g』的作法，因为 * 可以是 0 或多个重复前面的字符，而 . 是任意字符，所以： 『.* 就代表零个或多个任意字符』的意思啦！

```
[root@test root]# grep -n 'g.*g' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
14:The gd software is a library for drafting programs.
18:google is the best tools for search keyword.
19:goooooogle yes!
```

因为是代表 g 开头与 g 结尾，中间任意字符均可接受，所以，第 1 与第 14 行是可接受的喔！这个 .* 的 RE 表示任意字符是很常见的，希望大家能够理解并且熟悉！

再出一题，如果我想要找出『任意数字』的行列呢？因为仅有数字，所以就成为：

```
[root@test root]# grep -n '[0-9][0-9]*' regular_express.txt
5:However, this dress is about $ 3183 dollars.
15:You are the best is mean you are the no. 1.
```

虽然使用 grep -n '[0-9]' regular_express.txt 也可以得到相同的结果， 但鸟哥希望大家能够理解上面指令当中 RE 表示法的意义才好！

- 例题五、限定连续 RE 字符范围 {}
  在上个例题当中，我们可以利用 . 与 RE 字符及 * 来设定 0 个到无线多个重复字符， 那如果我想要限制一个范围区间内的重复字符数呢？举例来说，我想要找出两个到五个 o 的连续字符串，该如何作？这时候就得要使用到限定范围的字符 {} 了。 但因为 { 与 } 的符号在 shell 是

有特殊意义的，因此， 我们必须要使用跳脱字符 \ 来让他失去特殊意义才行。

至于 {} 的语法是这样的，假设我要找到两个 o 的字符串，可以是：

```
[root@test root]# grep -n 'o\{2\}' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
9:Oh! The soup taste good.
18:google is the best tools for search keyword.
19:goooooogle yes!
```

这样看似乎与 ooo* 的字符没有什么差异啊？因为第 19 行有多个 o 依旧也出现了！ 好，那么换个搜寻的字符串，假设我们要找出 g 后面接 2 到 5 个 o ，然后再接一个 g 的字符串， 他会是这样：

```
[root@test root]# grep -n 'go\{2,5\}g' regular_express.txt
18:google is the best tools for search keyword.
```

嗯！很好！第 19 行终于没有被取用了(因为 19 行有 6 个 o 啊！)。 那么，如果我想要的是 2 个 o 以上的 goooo....g 呢？除了可以是 gooo*g ，也可以是：

```
[root@test root]# grep -n 'go\{2,\}g' regular_express.txt
18:google is the best tools for search keyword.
19:goooooogle yes!
```

呵呵！就可以找出来啦～

重要特殊字符(characters)

经过了上面的几个简单的范例，我们可以将基础的正规表示法特殊字符汇整如下：

RE 字符	意义与范例
^word	待搜寻的字符串(word)在行首！ 范例：`grep -n '^#' regular_express.txt` 搜寻行首为 # 开始的那一行！
word$	待搜寻的字符串(word)在行尾！ 范例：`grep -n '!$' regular_express.txt` 将行尾为 ! 的那一行打印出来！
.	代表『任意一个』字符，一定是一个任意字符！ 范例：`grep -n 'e.e' regular_express.txt`

	搜寻的字符串可以是 (eve) (eae) (eee) (e e)， 但不能仅有 (ee) ！亦即 e 与 e 中间『一定』仅有一个字符，而空格符也是字符！
\	跳脱字符，将特殊符号的特殊意义去除！
	范例：grep -n \' regular_express.txt 搜寻含有单引号 ' 的那一行！
*	重复零个或多个的前一个 RE 字符
	范例：grep -n 'ess*' regular_express.txt 找出含有 (es) (ess) (esss) 等等的字符串，注意，因为 * 可以是 0 个，所以 es 也是符合带搜寻字符串。另外，因为 * 为重复『前一个 RE 字符』的符号，因此，在 * 之前必须要紧接着一个 RE 字符喔！例如任意字符则为 『.*』 ！
\{n,m\}	连续 n 到 m 个的『前一个 RE 字符』 若为 \{n\} 则是连续 n 个的前一个 RE 字符， 若是 \{n,\} 则是连续 n 个以上的前一个 RE 字符！
	范例：grep -n 'go\{2,3\}g' regular_express.txt 在 g 与 g 之间有 2 个到 3 个的 o 存在的字符串，亦即 (goog)(gooog)
[]	字符集合的 RE 特殊字符的符号
	[list] 范例：grep -n 'g[ld]' regular_express.txt 搜寻含有 (gl) 或 (gd) 的那一行～ 需要特别留意的是，在 [] 当中『谨代表一个待搜寻的字符』， 例如： a[afl]y 代表搜寻的字符串可以是 aay, afy, aly 亦即 [afl] 代表 a 或 f 或 l 的意思！ [ch1-ch2] 范例：grep -n '[0-9]' regular_express.txt 搜寻含有任意数字的那一行！需特别留意，在字符集合 [] 中的减号 - 是有特殊意义的，他代表两个字符之间的所有连续字符！但这个连续与否与 ASCII 编码有关， 因此，您的编码需要设定正确(在 bash 当中，需要确定 LANG 与 LANGUAGE 的变量是否正确！) 例如所有大写字符则为 [A-Z] [^] 范例：grep -n 'oo[^t]' regular_express.txt 搜寻的字符串可以是 (oog) (ood) 但不能是 (oot) ，那个 ^ 在 [] 内时，代表的意义是『反向选择』的意思～例如，我不要大写字符，则为 [^A-Z] ～但是，需要特别注意的是，如果以 grep -n [^A-Z] regular_express.txt 来搜寻，却发现该档案内的所有行都被列出，为什么？因为这个 [^A-Z] 是『非大写字符』的意思， 因为每一行均有非大写字符，例如第一行的 "Open Source" 就有 p,e,n,o.... 等等的小写字符， 以及双引号 (") 等字符，所以当然符合 [^A-Z] 的搜寻！

请特别留意的是，『正规表示法的特殊字符』 与一般在指令列输入指令的『万用字符』并不相同， 例如，在万用字符当中，* 代表的是 0 ~ 无限多个字符的意思，但是在正规表示法当中， * 则是重复 0 到多个的前一个 RE 字符的意思～使用的意义并不相同，不要搞混了！ (鸟哥我一开始摸正规表示法时就很容易

搞混！因为这里是新手最容易搞错的地方，特别小心啊！)

举例来说，不支持正规表示法的 ls 这个工具中，若我们使用 『ls -l * 』 代表的是任意档名的档案，而 『ls -l a* 』代表的是以 a 为开头的任何档名的档案， 但在正规表示法中，我们要找到含有以 a 为开头的档案，则必须要这样：(需搭配支持正规表示法的工具)

```
ls | grep -n '^a.*'
```

另外，例如万用字符的反向选择，为 [!range] ，至于正规表示法则是 [^range] 。 这样是否了解正规表示法与万用字符的差异啦？？

延伸正规表示法

事实上，一般读者只要了解基础型的正规表示法大概就已经相当足够了，不过，某些时刻， 为了要简化整个指令操作，了解一下使用范围更广的延伸型正规表示法的表示式，会更方便呢！ 举个简单的例子好了，在上节的例题三的最后一个例子中，我们要去除空白行与行首为 # 的行列， 使用的是

```
grep -v '^$' regular_express.txt | grep -v '^#'
```

需要使用到管线命令来搜寻两次！ 那么如果使用延伸型的正规表示法，我们可以简化为：

```
egrep -v '^$|^#' regular_express.txt
```

利用支持延伸型正规表示法的 egrep 与特殊字符 | 来区隔两组字符串，如此一来，是否方便很多呢？

这里必须要特别强调， grep 支持的是基础型的正规表示法，而 egrep 支持延伸正规表示法。 事实上，egrep 是 grep -E 的命令别名，为了方便使用，我们还是以 egrep 来跟 grep 区分吧！

熟悉了正规表示法之后，到这个延伸型的正规表示法，您应该也会想到，不就是多几个重要的特殊符号吗？^_^y 是的～所以，我们就直接来说明一下，延伸型正规表示法有哪几个特殊符号？

RE 字符	意义与范例
+	重复『一个或一个以上』的前一个 RE 字符
	范例：`egrep -n 'go+d' regular_express.txt` 搜寻 (god) (good) (goood)... 等等的字符串。那个 o+ 代表『一个以上的 o 』所以，上面的执行成果会将第 1, 9, 13 行列出来。
?	『零个或一个』的前一个 RE 字符
	范例：`egrep -n 'go?d' regular_express.txt` 搜寻 (gd) (god) 这两个字符串。 那个 o? 代表『空的或 1 个 o 』所以，上面的执行成果会将第 13, 14 行列出来。 有没有发现到，这两个案例('go+d' 与 'go?d')的结果集合与 'go*d' 相同？想想看，这是为什么喔！ ^_^
|	用或(or)的方式找出数个字符串
	范例：`egrep -n 'gd|good' regular_express.txt` 搜寻 gd 或 good 这两个字符串，注意，是『或』！ 所以，第 1,9,14 这三行都可以被打印出来喔！那如果还想要找出 dog 呢？就这样啊： egrep -n 'gd|good|dog' regular_express.txt
( )	找出『群组』字符串
	范例：`egrep -n 'g(la|oo)d' regular_express.txt` 搜寻 (glad) 或 (good) 这两个字符串，因为 g 与 d 是重复的，所以， 我

	就可以将 la 与 oo 列于 ( ) 当中，并以 | 来分隔开来，就可以啦！ 此外，这个功能还可以用来作为『多个重复群组』的判别喔！举例来说： echo 'AxyzxyzxyzxyzC' | egrep 'A(xyz)+C' 上面的例子当中，意思是说，我要找开头是 A 结尾是 C ，中间有一个以上的 "xyz" 字符串的意思～

以上这些就是延伸型的正规表示法的特殊字符。另外，要特别强调的是，那个 ! 在正规表示法当中并不是特殊字符， 所以，如果您想要查出来档案中含有 ! 与 > 的字行时，可以这样：

```
grep -n '[!>]' regular_express.txt
```

这样可以了解了吗？！常常看到有陷阱的题目写：『反向选择这样对否？ '[!a-z]'？』， 呵呵！是错的呦～要 '[^a-z] 才是对的！

格式化打印： printf

在很多时候，我们可能需要将输出的数据给他格式化输出的～ 举例来说，考试卷分数的输出，姓名与科目及分数之间，总是可以稍微作个比较漂亮的版面配置吧？ 例如我想要输出底下的样式：

```
Name     Chinese   English   Math    Average
DmTsai        80        60     92      77.33
VBird         75        55     80      70.00
Ken           60        90     70      73.33
```

分成五个字段，各个字段分配到正确的位置去！但是因为每个字段的原始数据其实并非是如此固定的， 而我就是想要如此表示出这些数据，此时，就得需要打印格式管理员 printf 的帮忙了！ printf 可以帮我们将资料输出的结果格式化，而且而支持一些特殊的字符～底下我们就来看看！

```
[root@linux ~]# printf '打印格式' 实际内容
参数：
关于格式方面的几个特殊样式：
       \a    警告声音输出
       \b    退格键(backspace)
       \f    清除屏幕 (form feed)
       \n    输出新的一行
       \r    亦即 Enter 按键
       \t    水平的 [tab] 按键
       \v    垂直的 [tabl] 按键
       \xNN  NN 为两位数的数字，可以转换数字成为字符。
关于 C 程序语言内，常见的变数格式
       %ns   那个 n 是数字， s 代表 string ，亦即多少个字符；
       %ni   那个 n 是数字， i 代表 integer ，亦即多少整数字数；
       %N.nf 那个 n 与 N 都是数字， f 代表 floating (浮点)，如果有小数字数，
             假设我共要十个位数，但小数点有两位，即为 %10.2f 啰！
范例：
```

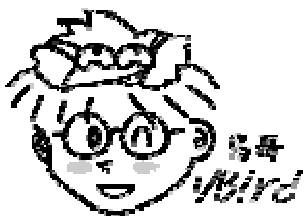
```
范例一：将刚刚上头的数据变成档案，仅列出姓名与成绩：(用 [tab] 分隔
[root@linux ~]# printf '%s\t %s\t %s\t %s\t %s\t \n' `cat printf.txt`
Name     Chinese         English         Math    Average
DmTsai   80      60      92      77.33
VBird    75      55      80      70.00
Ken      60      90      70      73.33
# 假设我将上面的档案存成 printf.txt 档案档名，则可利用上面的案例，
# 将每个单字中间以 [tab] 按键隔开。由上面的输出来看，虽然第二行以后是 OK 的，
# 但是第一行则因为某些单字长度较长，所以就无法对齐了！ 而 %s 表示以字符串 (string)
# 的方式来展现该内容。而每个内容则以 \t 即 [tab] 来隔开啊！

范例二：将上述资料关于第二行以后，分别以字符串、整数、小数点来显示：
[root@linux ~]# printf '%10s %5i %5i %5i %8.2f \n' `cat printf.txt |\
> grep -v Name`
    DmTsai    80    60    92    77.33
     VBird    75    55    80    70.00
       Ken    60    90    70    73.33
# 这个时候的输出可就有趣了！我将几个内容分成不同的数据格式来输出，
# 最有趣的应该是 %8.2f 这个项目了！我可以针对不同的小数字数来进行格式输出，
# 例如变成底下的样子时，您自己试看看，会是输出什么结果喔！
# printf '%10s %5i %5i %5i %8.1f \n' `cat printf.txt | grep -v Name`

范例三：列出数值 45 代表的字符为何？
[root@linux ~]# printf '\x45\n'
E
# 这东西也很好玩～他可以将数值转换成为字符，如果您会写 script 的话，
# 可以自行测试一下，由 20~80 之间的数值代表的字符是啥喔！ ^_^
```

printf 的使用相当的广泛喔！包括等一下后面会提到的 awk 以及在 C 程序语言当中使用的屏幕输出， 都是利用 printf 呢！鸟哥这里也只是列出一些可能会用到的格式而已， 有兴趣的话，可以自行多作一些测试与练习喔！ ^_^

Tips:
打印格式化这个 printf 指令，乍看之下好像也没有什么很重要的～ 不过，如果您需要自行撰写一些软件，需要将一些数据在屏幕上头漂漂亮亮的输出的话， 那么 printf 可也是一个很棒的工具喔！



---

sed 工具简介

在了解了一些正规表示法的基础应用之后，再来呢？呵呵～两个东西可以玩一玩的，那就是 sed 跟 awk 了！ 这两个家伙可是相当的有用的啊！举例来说，鸟哥写的 logfile.sh 分析登录文件的小程序， 绝大部分分析关键词的取用、统计等等，就是用这两个宝贝蛋来帮我完成的！ 那么你说，要不要玩一玩啊？！ ^_^

我们先来谈一谈 sed 好了，基本上， sed 可以分析 Standard Input (STDIN) 的数据， 然后将数据经过处

理后，再将他输出到 standrad out (STDOUT) 的一个工具。 至于处理呢？可以进行取代、删除、新增、撷取特定行等等的功能呢！很不错吧～ 我们先来了解一下 sed 的用法，再来聊他的用途好了！

```
[root@linux ~]# sed [-nefr] [动作]
参数：
-n  ：使用安静(silent)模式。在一般 sed 的用法中，所有来自 STDIN
      的数据一般都会被列出到屏幕上。但如果加上 -n 参数后，则只有经过
      sed 特殊处理的那一行(或者动作)才会被列出来。
-e  ：直接在指令列模式上进行 sed 的动作编辑；
-f  ：直接将 sed 的动作写在一个档案内， -f filename 则可以执行 filename 内的
      sed 动作；
-r  ：sed 的动作支持的是延伸型正规表示法的语法。(预设是基础正规表示法语法)

动作说明：   [n1[,n2]]function
n1, n2 ：不见得会存在，一般代表『选择进行动作的行数』，举例来说，如果我的动作
         是需要在 10 到 20 行之间进行的，则『 10,20[动作行为] 』

function 有底下这些咚咚：
a   ：新增， a 的后面可以接字符串，而这些字符串会在新的一行出现(目前的下一行)～
c   ：取代， c 的后面可以接字符串，这些字符串可以取代 n1,n2 之间的行！
d   ：删除，因为是删除啊，所以 d 后面通常不接任何咚咚；
i   ：插入， i 的后面可以接字符串，而这些字符串会在新的一行出现(目前的上一行)；
p   ：打印，亦即将某个选择的数据印出。通常 p 会与参数 sed -n 一起运作～
s   ：取代，可以直接进行取代的工作哩！通常这个 s 的动作可以搭配
      正规表示法！例如 1,20s/old/new/g 就是啦！
范例：

范例一：将 /etc/passwd 的内容列出，并且我需要打印行号，同时，请将第 2~5 行删除！
[root@linux ~]# nl /etc/passwd | sed '2,5d'
     1  root:x:0:0:root:/root:/bin/bash
     6  sync:x:5:0:sync:/sbin:/bin/sync
     7  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
.....(后面省略).....
# 看到了吧？因为 2-5 行给他删除了，所以显示的数据中，就没有 2-5 行啰～
# 另外，注意一下，原本应该是要下达 sed -e 才对，没有 -e 也行啦！
# 同时也要注意的是， sed 后面接的动作，请务必以 '' 两个单引号括住喔！
# 而，如果只要删除第 2 行，可以使用 nl /etc/passwd | sed '2d' 来达成，
# 至于第 3 到最后一行，则是 nl /etc/passwd | sed '3,$d' 的啦！

范例二：承上题，在第二行后(亦即是加在第三行)加上『drink tea?』字样！
[root@linux ~]# nl /etc/passwd | sed '2a drink tea'
     1  root:x:0:0:root:/root:/bin/bash
     2  bin:x:1:1:bin:/bin:/sbin/nologin
drink tea
```

```
     3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
# 嘿嘿！在 a 后面加上的字符串就已将出现在第二行后面啰！那如果是要在第二行前呢？
# nl /etc/passwd | sed '2i drink tea' 就对啦！

范例三：在第二行后面加入两行字，例如『Drink tea or .....』『drink beer?』
[root@linux ~]# nl /etc/passwd | sed '2a Drink tea or ......\
> drink beer ?'
     1  root:x:0:0:root:/root:/bin/bash
     2  bin:x:1:1:bin:/bin:/sbin/nologin
Drink tea or ......
drink beer ?
     3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
# 这个范例的重点是，我们可以新增不只一行喔！可以新增好几行～
# 但是每一行之间都必须要以反斜线 \ 来进行新行的增加喔！所以，上面的例子中，
# 我们可以发现在第一行的最后面就有 \ 存在啦！那是一定要的喔！

范例四：我想将第 2-5 行的内容取代成为『No 2-5 number』呢？
[root@linux ~]# nl /etc/passwd | sed '2,5c No 2-5 number'
     1  root:x:0:0:root:/root:/bin/bash
No 2-5 number
     6  sync:x:5:0:sync:/sbin:/bin/sync
# 没有了 2-5 行，嘿嘿嘿嘿！我们要的数据就出现啦！

范例五：仅列出第 5-7 行
[root@linux ~]# nl /etc/passwd | sed -n '5,7p'
     5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
     6  sync:x:5:0:sync:/sbin:/bin/sync
     7  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
# 为什么要加 -n 的参数呢？您可以自行下达 sed '5,7p' 就知道了！(5-7 行会重复输出)
# 有没有加上 -n 的参数时，输出的数据可是差很多的喔！

范例六：我们可以使用 ifconfig 来列出 IP ，若仅要 eth0 的 IP 时？
[root@linux ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:51:FD:52:9A:CA
          inet addr:192.168.1.12  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::250:fcff:fe22:9acb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
.....(以下省略).....
# 其实，我们要的只是那个 inet addr:..那一行而已，所以啰，利用 grep 与 sed 来捉
[root@linux ~]# ifconfig eth0 | grep 'inet ' | sed 's/^.*addr://g' | \
> sed 's/Bcast.*$//g'
# 您可以将每个管线 (|) 的过程都分开来执行，就会晓得原因啰！
# 去头去尾之后，就会得到我们所需要的 IP 亦即是 192.168.1.12 啰～
```

```
范例七：将 /etc/man.config 档案的内容中，有 MAN 的设定就取出来，但不要说明内容。
[root@linux ~]# cat /etc/man.config | grep 'MAN'| sed 's/#.*$//g' | \
> sed '/^$/d'
# 每一行当中，若有 # 表示该行为批注，但是要注意的是，有时候，
# 批注并不是写在第一个字符，亦即是写在某个指令后方，如底下的模样：
# 『shutdown -h now # 这个是关机的指令』，批注 # 就在指令的后方了。
# 因此，我们才会使用到将 #.*$ 这个正规表示法！
```

总之，这个 sed 不错用啦！而且很多的 shell script 都会使用到这个指令的功能～ sed 可以帮助系统管理员管理好日常的工作喔！要仔细的学习呢！

---

awk 工具简介

相较于 sed 常常作用于一整个行的处理， awk 则比较倾向于一行当中分成数个『字段』来处理。 因此，awk 相当的适合处理小型的数据数据处理呢！awk 通常运作的模式是这样的：

```
[root@linux ~]# awk '条件类型 1{动作 1} 条件类型 2{动作 2} ...' filename
```

awk 可以处理后续接的档案，也可以读取来自前个指令的 standard output 。 但如前面说的， awk 主要是处理『每一行的字段内的数据』，而预设的『字段的分隔符为 "空格键" 或 "[tab]键" 』！举例来说，我们用 last 可以将登入者的数据取出来， 结果如下所示：

```
[root@linux ~]# last
dmtsai   pts/0        192.168.1.12     Mon Aug 22 09:40   still logged in
root     tty1                          Mon Aug 15 11:38 - 11:39  (00:01)
reboot   system boot  2.6.11           Sun Aug 14 18:18         (7+15:41)
dmtsai   pts/0        192.168.1.12     Fri Aug 12 12:07 - 12:08  (00:01)
```

若我想要取出账号与登入者的 IP ，且账号与 IP 之间以 [tab] 隔开，则会变成这样：

```
[root@linux ~]# last | awk '{print $1 "\t" $3}'
dmtsai  192.168.1.12
root    Mon
reboot  boot
dmtsai  192.168.1.12
```

因为不论哪一行我都要处理，因此，就不需要有 "条件类型" 的限制！我所想要的是第一栏以及第三栏， 但是，第二行及第三行的内容怪怪的～这是因为数据格式的问题啊！所以啰～使用 awk 的时候，请先确认一下您的数据当中，如果是连续性的数据，请不要有空格或 [tab] 在内，否则，就会像这个例子这样，会发生误判喔！

另外，由上面这个例子您也会知道，在每一行的每个字段都是有变量名称的，那就是 $1, $2... 等变量名称，以上面的例子来说， dmtsai 是 $1 ，因为他是第一栏嘛！至于 192.168.1.12 是第三栏， 所以他就是 $3 啦！后面以此类推～呵呵！还有个变数喔！那就是 $0 ，$0 代表『一整列资料』的意思～ 以上面的例子来说，第一行的 $0 代表的就是『dmtsai pts/0.... 』那一行啊！ 由此可知，刚刚上面四行当中，整个 awk 的处理流程是：

1. 读入第一行，并将第一行的资料填入 $0, $1, $2.... 等变数当中；
2. 依据 "条件类型" 的限制，判断是否需要进行后面的 "动作"；
3. 做完所有的动作与条件类型；
4. 若还有后续的『行』的数据，则重复上面 1~3 的步骤，直到所有的数据都读完为止。

经过这样的步骤，您会晓得， awk 是『以行为一次处理的单位』， 而『以字段为最小的处理单位』。好了，那么 awk 怎么知道我到底这个数据有几行？有几栏呢？这就需要 awk 的内建变量的帮忙啦～

变量名称	代表意义
NF	每一行 ($0) 拥有的字段总数
NR	目前 awk 所处理的是『第几行』数据
FS	目前的分隔字符，预设是空格键

我们继续以上面例子来做说明，如果我想要列出每一行的账号，并且列出目前处理的行数， 并且说明，该行有多少字段，则可以这样 (注意， awk 后续的所有动作以 ' 括住， 所以，内容如果想要以 print 打印时，记得，非变量的文字部分，包含上一小节 printf 提到的格式中，都需要使用双引号来定义出来喔！)

```
[root@linux ~]# last | awk '{print $1 "\t lines: " NR "\t columes: " NF}'
dmtsai   lines: 1        columes: 10
root     lines: 2        columes: 9
reboot   lines: 3        columes: 9
dmtsai   lines: 4        columes: 10
```

这样可以了解 NR 与 NF 的差别了吧？好了，底下来谈一谈所谓的 "条件类型" 了吧！

---

awk 的逻辑运算字符

既然有需要用到 "条件" 的类别，自然就需要一些逻辑运算啰～例如底下这些：

运算单元	代表意义
>	大于
<	小于
>=	大于或等于
<=	小于或等于
==	等于
!=	不等于

值得注意的是那个 == 的符号，因为在『逻辑运算』上面， 就是所谓的大于、小于、等于等等的判断式上面，我们习惯上是以 == 来表示，而如果是直接给予一个值，例如变量设定时，就直接使用 = 而已。 好了，我们实际来运用一下逻辑判断吧！举例来说，在 /etc/passwd 当中是以冒号 ":" 来作为字段的分隔，那假设我要查阅，第三栏小于 10 以下的数据，并且仅列出账号与第三栏， 那么可以这样做：

```
[root@linux ~]# cat /etc/passwd | \
> awk '{FS=":"} $3 < 10 {print $1 "\t " $3}'
root:x:0:0:root:/root:/bin/bash
bin       1
daemon    2
......(以下省略)......
```

有趣吧！不过，怎么第一行没有正确的显示出来呢？这是因为我们读入第一行的时候， 那些变数 $1, $2... 预设还是以空格键为分隔的，所以虽然我们定义了 FS=":" 了， 但是却仅能在第二行后才开始生效。那么怎么办呢？我们可以预先设定 awk 的变量啊！ 利用 BEGIN 这个关键词喔！这样做：

```
[root@linux ~]# cat /etc/passwd | \
> awk 'BEGIN {FS=":"} $3 < 10 {print $1 "\t " $3}'
root      0
bin       1
daemon    2
......(以下省略)......
```

很有趣吧！而除了 BEGIN 之外，我们还有 END 呢！另外，如果要用 awk 来进行『计算功能』呢？以底下的例子来看， 假设我有一个薪资数据表，内容是这样的：

```
Name     1st      2nd      3th
VBird    23000    24000    25000
DMTsai   21000    20000    23000
Bird2    43000    42000    41000
```

如何帮我计算每个人的总额呢？而且我还想要格式化输出喔！ 你可以将上面的数据储存成一个名称为 pay.txt 的档案，则：

```
[root@linux ~]# cat pay.txt | \
> awk 'NR==1{printf "%10s %10s %10s %10s %10s\n",$1,$2,$3,$4,"Total" }
NR>=2{total = $2 + $3 + $4
printf "%10s %10d %10d %10d %10.2f\n", $1, $2, $3, $4, total}'
      Name        1st        2nd        3th      Total
     VBird      23000      24000      25000   72000.00
    DMTsai      21000      20000      23000   64000.00
     Bird2      43000      42000      41000  126000.00
```

上面的例子有几个重要事项应该要先说明的：

- 所有的动作，亦即在 {} 内的动作，如果有需要多个指令辅助时，可利用分号『;』间隔， 或者直接以 [Enter] 按键来隔开每个指令，例如上面的 NR>=2 后面接的动作， 利用 total = ... 那个指令来指定加总，而后续则以 printf 来格式化输出！
- 逻辑运算当中，如果是『等于』的情况，则务必使用两个等号『==』！
- 格式化输出时，在 printf 的格式设定当中，务必加上 \n ，才能进行分行！
- 与 bash shell 的变量不同，在 awk 当中，变量可以直接使用，不需加上 $ 符号。

利用 awk 这个玩意儿，就可以帮我们处理很多日常工作了呢！真是好用的很～ 此外， awk 的输出格式

当中，常常会以 printf 来辅助，所以， 最好您对 printf 也稍微熟悉一下比较好啦！另外， awk 的动作内 {} 也是支持 if (条件) 的喔！ 举例来说，上面的指令可以修订成为这样：

```
[root@linux ~]# cat pay.txt | \
> awk '{if(NR==1) printf "%10s %10s %10s %10s %10s\n",$1,$2,$3,$4,"Total"}
NR>=2{total = $2 + $3 + $4
printf "%10s %10d %10d %10d %10.2f\n", $1, $2, $3, $4, total}'
```

你可以仔细的比对一下上面两个输入有啥不同～从中去了解两种语法吧！ 我个人是比较倾向于使用第一种语法，因为会比较有统一性啊！ ^_^

除此之外， awk 还可以帮我们进行循环计算喔！真是相当的好用！ 不过，那属于比较进阶的单独课程了，我们这里就不再多加介绍。如果您有兴趣的话， 可以到中研院的网站查询喔：http://phi.sinica.edu.tw/aspac/reports/94/94011/ ，鸟哥这里也有一份 pdf 档的备份：http://linux.vbird.org/linux_basic/0330regularex/awk.pdf。 您可以自行参阅一下该文章的内容，里头可以好好的查阅一下关于数组与循环方面的介绍， 我认为该文章写的很棒喔！该介绍的都介绍了！很好～我喜欢～^_^

文件数据比对与打印的相关功能

正规表示法是相当有用的工具，当然，那个 sed 还有 awk 也是很棒的工具程序， 不过，除此之外，我们其实还有很多可以使用的工作来处理文件数据喔！ 举例来说，假如我有两个档案，一个档案是原始档，一个则是经过一些时间累积处理后的档案， 我想要知道这两个档案之间的差别，该如何运用正规表示法？呼呼～可能要透过所谓的循环来一行一行比对检查呢～ 但是，我们可以透过 Linux 提供的 diff 及 cmp 指令来进行比对即可喔！很棒的啊！

档案比对

什么时候会用到档案的比对啊？通常是『同一个软件包的不同版本之间，比较设定档与原始档的差异』， 所以啰，很多时候所谓的档案比对，通常是用在 ASCII 纯文字文件的比对上的！ 那么比对档案的指令有哪些？最常见的就是 diff 啰！

- diff

diff 就是用在比对两个档案之间的差异的，一般是用在 ASCII 纯文字文件的比对上。 我们先预处理一下一个档案好了。假设我要将 /etc/passwd 的内容，将第四行删除， 第六行则取代成为『no six line』，新的档案放置到 /tmp/test 里面， 那么应该怎么做？

```
[root@linux ~]# mkdir -p /tmp/test
[root@linux ~]# cat /etc/passwd | \
> sed -e '4d' -e '6c no six line' > /tmp/test/passwd
# 注意一下， sed 后面如果要接超过两个以上的动作时，每个动作前面得加 -e 才行！
```

接下来讨论一下关于 diff 的用法吧！

```
[root@linux ~]# diff [-bBi] from-file to-file
```

```
参数：
from-file ：一个档名，作为原始比对档案的档名；
to-file   ：一个档名，作为目的比对档案的档名；
注意，from-file 或 to-file 可以 - 取代，那个 - 代表『Standard input』之意。

-b  ：忽略一行当中，仅有多个空白的差异(例如 "about me" 与 "about     me" 视为相同
-B  ：忽略空白行的差异。
-i  ：忽略大小写的不同。
范例：

范例一：比对 /tmp/test/passwd 与 /etc/passwd 的差异：
[root@linux ~]# diff /etc/passwd /tmp/test/passwd
4d3    <==这里是说，左边档案(/etc/passwd)第四行被删除 (d)
< adm:x:3:4:adm:/var/adm:/sbin/nologin
6c5    <==这里是说，左边档案的第六行被取代成右边档案(/tmp/test/passwd)的第五行
< sync:x:5:0:sync:/sbin:/bin/sync
---
> no six line
# 很聪明吧！用 diff 就把我们刚刚的处理给比对完毕了！
```

用 diff 比对档案真的是很简单喔！另外， diff 也可以比对整个目录下的差异喔！ 举例来说，我们将两个目录比对一下：

```
[root@linux ~]# diff /etc /tmp/test
......(前面省略).....
Only in /etc: paper.config
diff /etc/passwd /tmp/test/passwd
4d3
< adm:x:3:4:adm:/var/adm:/sbin/nologin
6c5
< sync:x:5:0:sync:/sbin:/bin/sync
---
> no six line
Only in /etc: passwd-
......(后面省略).....
```

我们的 diff 很聪明吧！还可以比对不同目录下的相同文件名的内容， 这样真的很方便喔～

- cmp

相对于 diff 的广泛用途， cmp 似乎就用的没有这么多了～ cmp 主要也是在比对两个档案，他主要利用『位』单位去比对，因此， 当然也可以比对 binary file 啰～(还是要再提醒喔， diff 主要是以『行』为单位比对， cmp 则是以『位』为单位去比对，这并不相同！)

```
[root@linux ~]# cmp [-s] file1 file2
参数：
```

```
-s  ：将所有的不同点的位处都列出来。因为 cmp 预设仅会输出第一个发现的不同点。
范例：

范例一：用 cmp 比较一下 /etc/passwd 与 /tmp/test/passwd
[root@linux ~]# cmp /etc/passwd /tmp/test/passwd
/etc/passwd /tmp/test/passwd differ: byte 106, line 4
```

看到了吗？第一个发现的不同点在第四行，而且位数是在第 106 个位处！ 这个 cmp 也可以用来比对 binary 啦！ ^_^

- patch

patch 这个指令与 diff 可是有密不可分的关系啊！我们前面提到，diff 可以用来分辨两个版本之间的差异，举例来说，刚刚我们所建立的 /tmp/test/passwd 与 /etc/passwd 就是两个不同版本之间的档案。那么，如果要『升级』呢？ 就是『将旧的档案升级成为新的档案』时，应该要怎么做呢？ 举例来说，我们可以这样做测试：

```
[root@linux ~]# mkdir /tmp/old; cp /etc/passwd /tmp/old
[root@linux ~]# mkdir /tmp/new; cp /tmp/test/passwd /tmp/new
[root@linux ~]# cd /tmp ; diff -Naur old/ new/ > test.patch
```

此时，在 /tmp/test.patch 档案之中，就记录了新旧的档案之间的差异， 对了！您必须要了解的是，用 diff 制作这个档案时，旧的档案必须是在前面，亦即是 diff oldfile newfile 才行喔！此外，新旧档案的『相对目录位置』最好也是一样比较好喔！ OK！那么如何将旧的内容 (/tmp/old/passwd) 更新到新版 (/tmp/new/passwd) 的内容呢？ 简单的说，可以用这样：

```
[root@linux ~]# patch -pN < patch_file
参数：
-p  ：后面可以接『取消几层目录』的意思。
范例：

范例一：将刚刚制作出来的 patch file 用来更新旧版数据
[root@linux ~]# cd /tmp/old
[root@linux ~]# patch -p1 < /tmp/test.patch
patching file passwd
# 为什么这里会使用 -p1 呢？因为我们在比对新旧版的数据时，是在 /tmp 底下，
# 而实际的数据是在 /tmp/old 里面，因此，当我们进入到 /tmp/old 时，
# 再查阅 /tmp/test.patch 的第一行如下：
# diff -Naur old/passwd new/passwd (用 head -n 1 /tmp/test.patch)
# 发现到，我们所在的目录其实是 old 里面，所以，就必须要减去一层目录。
```

更详细的 patch 用法我们会在后续的第五章跟大家介绍， 这里仅是介绍给您，呵呵！我们可以利用 diff 来比对两个档案之间的差异， 更可进一步利用这个功能来制作修补档案 (patch file) ，让大家更容易进行比对与升级呢！ 很不赖吧！ ^_^

档案打印准备： pr

如果您曾经使用过一些图形接口的文字处理软件的话，那么很容易发现， 当我们在打印的时候，可以同时选择与设定每一页打印时的标头吧！ 也可以设定页码呢！那么，如果我是在 Linux 底下打印纯文字文件呢 可不可以具有标题啊？可不可以加入页码啊？呵呵！当然可以啊！ 使用 pr 就能够达到这个功能了。不过， pr 的参数实在太多了， 我也说不完，一般来说，我都仅使用最简单的方式来处理而已。 举例来说，如果想要打印 /etc/man.config 呢？

```
[root@linux ~]# pr /etc/man.config


2003-02-10 23:20                /etc/man.config                 Page 1


#
# Generated automatically from man.conf.in by the
# configure script.
.....以下省略......
```

上面特殊字体那一行呢，其实就是使用 pr 处理后所造成的标题啦～ 标题中会有『档案时间』、『档案档名』及『页码』三大项目。 更多的 pr 使用，请参考 pr 的说明啊！ ^_^

重点回顾

- 使用 grep 或其它工具进行正规表示法的字符串比对时，因为编码的问题会有不同的状态，因此，您最好将 LANG 及 LANGUAGE 等变量设定为 C 或者是 en 等英文语系！
- 正规表示法 (Regular Expression) 的用途主要是用来做为『搜寻』字符串之用，还可以用来过滤特殊讯息等用途；
- 由于严谨度的不同，正规表示法之上还有更严谨的延伸正规表示法；
- 正规表示法的处理方式，经常是以『整行』或称为『整段』来进行处理的；
- grep 与 egrep 在正规表示法里面是很常见的两支程序，其中， egrep 支持更严谨的正规表示法的语法；

参考资源

- 洪朝贵老师的网页：http://www.cyut.edu.tw/~ckhung/olbook/gnulinux/regexp.shtml
- PCRE 官方网站：http://www.perldoc.com/perl5.8.0/pod/perlre.html
- 龙门少尉的窝：http://main.rtfiber.com.tw/~changyj/
- Study Area：http://www.study-area.org/linux/system/linux_shell.htm
- 中研院计算中心 ASPAC 计划之 awk 程序介绍：http://phi.sinica.edu.tw/aspac/reports/94/94011/ 鸟哥备份：http://linux.vbird.org/linux_basic/0330regularex/awk.pdf
- 中研院计算中心 ASPAC 计划之 sed 程序介绍：http://phi.sinica.edu.tw/aspac/reports/96/96005/

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 我想要知道某个档案里面含有 boot 的字眼，而这个档案在 /etc/ 底下，我要如何找出这个档案？

  既然知道有这个字眼那就好办了！可以直接下达：

  ```
  grep boot /etc/*
  ```

- 我想要知道，在 /etc 底下，只要含有 XYZ 三个字符的任何一个字符的那一行就列出来，要怎样进行？

  『只要』含有 X 或 Y 或 Z 就将该行列出来，因此，我们的范围很很广啦！这个时候就必需要使用到 [] 这个咚咚！还记得中括号的用途吗？那就是『在中括号里面谨代表一个字符而已！』 而这个中括号是一个『代表』，可以是一串字也可以是几个不连续的字！这里我们仅需要 XYZ 其中任何一个，所以可以这样写：

  ```
  grep [XYZ] /etc/*
  ```

  则只要在每一行当中，只要发现 X 或 Y 或 Z 任何一个，就会将他印出来！这个与 grep XYZ /etc/* 是『完全不一样』的！请仔细的思考一下ㄟ！

- 我想要找出在 /etc 底下，档案内容含有 * 的文件名称？

  由于 * 是特殊字符，在变量的订定法则里面曾经提过要将特殊字符移除，需要使用跳脱字符，亦即是 \ 符号，所以我可以这样下达指令：

  ```
  grep \* /etc/*
  ```

学习 Shell Scripts

最近更新日期：2005/08/29

如果您真的很想要走信息这条路，并且想要好好的管理好属于您的主机，那么，别说鸟哥不告诉您， Shell Scripts 真的是必须要学习的一项课题呢！基本上， shell script 有点像是早期的批次档， 亦即是将一些指令汇整起来一次执行，但是 Shell script 拥有更强大的功能，那就是， 他可以进行类似程序 (program) 的撰写，并且，不需要经过编译 (compiler) 就能够执行， 真的很方便。加上，我们可透过 shell script 来简化我们日常的工作管理， 而且，整个 Linux 环境中，一些服务 (services) 的启动都是透过 shell script 的，如果您对于 script 不了解，嘿嘿！发生问题时，可真是会求助无门喔！ 所以，好好的学一学他吧！

什么是 Shell scripts ？

这个有趣的问题赶紧来回答看看，什么是 shell script 呢？ shell 我们在 认识 bash 当中已经提过了，那是一个文字接口底下让我们与系统沟通的一个工具接口，那么 script 是啥？ 字面上的意义， script 是『脚本、剧本』的意思。整句话是说， shell script 是针对 shell 所写的『剧本！』 什么东西啊？ 呵呵！其实， shell script 是利用 shell 的功能所写的一个『程序 (program)』，这个程序是使用纯文字文件，将一些 shell 的语法与指令写在里面， 搭配正规表示法、管线命令与数据流重导向等功能，以达到我们所想要的处理目的。

所以，简单的说， shell script 就像是早期 DOS 年代的批次檔 (.bat) ，最简单的功能就是将许多指令汇整写在一起， 让使用者很轻易的就能够 one touch (执行一个档案 "shell script" ，就能够一次执行多个指令)， 而， shell script 更提供数组、循环、条件与逻辑判断等重要功能，让使用者也可以直接以 shell 来撰写程序，而不必使用类似 C 程序语言等传统程序撰写的语法呢！

那，这么说您可以了解了吗？是的！ shell script 可以简单的被看成是批次档， 也可以被说成是一个程序语言，且这个程序语言由于都是利用 shell 与相关工具指令， 所以不需要编译即可执行，且拥有不错的除错 (debug) 工具，所以，他可以帮助系统管理员快速的管理好主机。

干嘛学习 shell scripts?

这是个好问题，我又干嘛一定要学 shell script ？我又不是信息人，没有写程序的概念， 那我干嘛还要学 shell script 呢？不要学可不可以啊？呵呵～如果 Linux 对您而言， 您只是想要『会用』而已，那么，不需要学 shell script 也还无所谓，这部分先给他跳过去， 等到有空的时候，再来好好的瞧一瞧。但是，如果您是真的想要玩清楚 Linux 的来龙去脉， 那么 shell script 就不可不知，为什么呢？因为：

- 自动化管理的重要依据：

不用鸟哥说您也知道，管理一部主机真不是件简单的事情，每天要进行的任务就有： 查询登录档、追踪流量、监控使用者使用主机状态、主机各项硬设备状态、 主机软件更新查询、更不要说得应付其它使用者的突然要求了。而这些工作， 您想要自行手动处理，还是写个简单的程序来帮您每日自动处理分析，若有问题才通知您呢？ 当然是让系统自动工作比较好，对吧！呵呵～这就得要良好的 shell script 来帮忙的啦！

- 追踪与管理系统的重要工作：

虽然我们还没有提到服务启动的方法，不过，这里可以先提一下，我们 Linux 系统的服务 (services) 启动的接口，在 /etc/init.d/ 这个目录下，所有的档案都是 scripts ； 另外，包括开机 (booting) 过程也都是利用 shell script 来帮忙搜寻系统的相关设定数据， 然后再代入各个服务的设定参数啊！举例来说，如果我们想要重新启动系统登录文件， 可以使用：『/etc/init.d/syslogd restart』，那个 syslogd 档案就是 script 啦！ 另外，我曾经在某一代的 FC 上面发现，启动 MySQL 这个数据库服务时，确实是可以启动的， 但是屏幕上却老是出现『failure』，后来才发现，原来是启动 MySQL 那个 script 会主动的以『空的密码』去尝试登入 MySQL ，但我修改过 MySQL 的密码啰～当然就登入失败～ 后来改了改 script ，就略去这个问题啦！如此说来， script 确实是需要学习的啊！

- 简单入侵侦测功能：

当我们的系统有异状时，大多会将这些异状记录在系统记录器，也就是我们常提到的『系统登录文件』， 那么我们可以在固定的几分钟内主动的去分析系统登录文件，若察觉有问题，就立刻通报管理员， 或者是立刻加强防火墙的设定规则，如此一来，您的主机可就能够达到『自我保护』的聪明学习功能啦～ 举例来说，我们可以通过 shell script 去分析『当该封包尝试几次还是联机失败之后，就予以抵挡住该 IP』之类的举动，例如鸟哥写过一个关于抵挡砍站软件的 shell script ， 就是用这个想法去达成的呢！

- 连续指令单一化：

其实，对于新手而言， script 最简单的功能就是：『汇整一些在 command line 下达的连续指令，将他写入 scripts 当中，而由直接执行 scripts 来启动一连串的 command line 指令输出入！』例如： 防火墙连续规则 (iptables)，开机加载程序的项目 (就是在 /etc/rc.d/rc.local 里头的数据) ，等等都是相似的功能啦！ 其实，说穿了，如果不考虑 program 的部分，那么 scripts 也可以想成，仅是帮我们把一大串的指令汇整在一个档案里面， 而直接执行该档案就可以执行那一串又臭又长的指令段！就是这么简单啦！

- 简易的数据处理：

由前一章 正规表示法 的 awk 程序说明中， 您可以发现， awk 可以用来处理简单的数据数据呢！例如薪资单的处理啊等等的。 shell script 的功能更强大，例如鸟哥曾经用 shell script 直接处理数据数据

的比对啊， 文字数据的处理啊等等的，撰写方便，速度又快(因为在 Linux 效能较佳)， 真的是很不错用的啦！

- 跨平台支持与学习历程较短：

几乎所有的 Unix Like 上面都可以跑 shell script ，连 MS Windows 系列也有相关的仿真器可以用， 此外， shell script 的语法是相当亲和的，看都看的懂得文字，而不是机器码， 很容易学习～这些都是您可以加以考虑的学习点啊！

上面这些都是您考虑学习 shell script 的特点～此外， shell script 还可以简单的以 vi 来直接编写，实在是很方便的好东西！所以，还是建议您学习一下啦。

不过，虽然 shell script 号称是程序 (program) ，但实际上， shell script 处理数据的速度上是不太够的。因为 shell script 用的是外部的指令与 bash shell 的一些预设工具，所以，他常常会去呼叫外部的函式库，因此，运算速度上面当然比不上传统的程序语言。 所以啰， shell script 用在系统管理上面是很好的一项工具，但是用在处理大量数值运算上， 就不够好了～而且还很麻烦，因为：Shell scripts 的速度较慢， 且使用的 CPU 资源较多，造成主机资源的分配不良。还好， 我们确实很少看到利用 shell script 在进行大量数据运算的，所以，不必担心的啦！

第一支 script 的撰写与执行

如同前面讲到的， shell script 其实就是纯文字文件 (ASCII) ，我们可以编辑这个档案， 然后让这个档案来帮我们一次执行多个指令，或者是利用一些运算与逻辑判断来帮我们达成某些功能。 所以啦，要编辑这个档案的内容时，当然就需要具备有 bash shell 指令下达的相关认识。 我们说过，要下达指令需要注意的事项在 bash 章节内已经提过， 在 shell script 的撰写同样需要用到这些注意事项的：

1. 如同前面 bash command 提到的，指令与参数间的多个空白会被忽略掉；
2. 而空白行也将被忽略掉！，并且 [tab] 也是不会被理会的！
3. 如果读取到一个 Enter 符号 (CR)），就尝试开始执行该行命令；
4. 至于如果一行的内容太多，则可以使用 \[Enter] 来延伸至下一行；
5. 此外，使用最多的 # 可做为批注！任何加在 # 后面的字，将全部被视为批注文字而被忽略！

如此一来，我们在 script 内所撰写的程序，就会被一行一行的执行。好了，那么这个程序假设文件名是 shell.sh 好了，如何执行这个档案？很简单，可以有底下几个方法：

- 将 shell.sh 加上可读与执行 (rx) 的权限，然后就能够以 ./shell.sh 来执行了；
- 直接以 sh shell.sh 的方式来直接执行即可。

反正重点就是要让那个 shell.sh 内的指令可以被执行的意思啦！咦！那我为何需要使用 ./shell.sh 来下达指令？ 还记得我们在 bash 里面一直强调的，指令是否能够被执行与 PATH 这个环境变量有关， 所以，要执行『目前这个目录下的某个档案』就需要加上 ./ 这个目录啦！另外，其实您也可以将 shell.sh 放在您家目录下的 ~/bin 这个目录中，然后利用 PATH="$PATH":~/bin 的设定， 嘿嘿，就能够直接执行您的 script 啰～ ^_^

那，为何 sh shell.sh 也可以执行呢？这是因为 /bin/sh 其实就是 /bin/bash ， 使用 sh shell.sh 亦

即告诉系统，我想要直接以 bash 的功能来执行 shell.sh 这个档案内的相关指令的意思。 而我们也可以利用 sh 的参数，如 -n 及 -x 来检查与追踪 shell.sh 的语法是否正确呢！ ^_^

- 撰写第一支 script

不论是那个门派，要学武功要从扫地做起，那么要学程序呢？呵呵，肯定是由『秀出 Hello World！』 这个字眼开始的！OK！那么鸟哥就先写一支 script 给大家瞧一瞧：

```
[root@linux ~]# mkdir scripts; cd scripts
[root@linux scripts]# vi sh01.sh
#!/bin/bash
# Program:
#       This program is used to show "Hello World !" in screen.
# History:
# 2005/08/23        VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH
echo -e "Hello World ! \a \n"
exit 0
```

在我们这个章节当中，请将所有的撰写的 script 放置到您家目录的 ~/scripts 这个目录内， 比较好管理啦！上面的写法当中，我主要将整个程序的撰写分成数段，大致是这样：

1. 第一行 #!/bin/bash 在宣告这个 script 使用的 shell 名称：
   因为我们使用的是 bash ，所以，必须要以『 #!/bin/bash 』来宣告这个档案内的语法使用 bash 的语法！那么当这个程序被执行时，他就能够加载 bash 的相关环境设定档， 并且执行 bash 来使我们底下的指令能够执行！这很重要的！(在很多状况中，如果没有设定好这一行， 那么该程序很可能会无法执行，因为系统可能无法判断该程序需要使用什么 shell 来执行啊！)

2. 程序内容的宣告：
   整个 script 当中，除了第一行的 #! 是用来宣告 shell 的之外，其它的 # 都是『批注』用途！所以上面的程序当中，第二行以下就是用来说明整个程序的状态。一般来说， 建议您一定要养成说明该 script 的：1. 内容与功能； 2. 版本信息； 3. 作者与联络方式； 4. 建檔日期；5. 历史纪录 等等。这将有助于未来程序的改写与 debug 呢！

3. 主要环境变量的宣告：
   建议务必要将一些重要的环境变量设定好，鸟哥个人认为， PATH 是当中最重要的！ 如此一来，则可让我们这支程序在进行时，可以直接下达指令， 而不必写绝对路径呢！比较好啦！

4. 主要程序部分
   就将主要的程序写好即可！在这个例子当中，就是 echo 那一行啦！

5. 执行成果告知
   是否记得我们在 bash 里面要讨论一个指令的执行成功与否，可以使用 $? 这个变量来观察～ 那么我们也可以利用 exit 这个指令来让程序中断，并且回传一个数值给系统。 在我们这个例子当中，我使用 exit 0 ，这代表离开 script ，并且回传一个 0 给系统， 所以我执行完这个 script

后，若接着下达 echo $? 则可得到 0 的值喔！更聪明的读者应该也知道了，呵呵！利用这个 exit n 的功能，我们还可以自订错误讯息， 让这支程序变得更加的 smart 呢！

接下来执行看看结果是怎样吧？

```
[root@linux scripts]# sh sh01.sh
Hello World !
```

您会看到屏幕是这样，而且应该还会听到『咚』的一声，为什么呢？还记得前一章提到的 printf 吧？用 echo 接着那些特殊的按键也可以发生同样的事情～ 不过， echo 必须要加上 -e 的参数才行！ 呵呵！在您写完这个小 script 之后，您就可以大声的说：『我也会写程序了』！哈哈！ 很简单有趣吧～ ^_^

另外，你也可以利用：『chmod a+x sh01.sh; ./sh01.sh』来执行这个 script 的呢！

撰写 shell script 的良好习惯建立

一个良好习惯的养成是很重要的～大家在刚开始撰写程序的时候，最容易忽略这部分， 认为程序写出来就好了，其它的不重要。其实，如果程序的说明能够更清楚， 那么对您自己是有很大的帮助的。

举例来说，鸟哥自己为了自己的需求，曾经撰写了不少的 script 来帮我进行主机 IP 的侦测啊、 登录档分析与管理啊、自动上传下载重要设定档啊等等的，不过，早期就是因为太懒了， 管理的主机又太多了，常常同一个程序在不同的主机上面进行更改，到最后，到底哪一支才是最新的都记不起来， 而且，重点是，我到底是改了哪里？？为什么做那样的修改？都忘的一乾二净～真要命～

所以，后来鸟哥在写程序的时候，通常会比较仔细的将程序的设计过程给他记录下来， 而且还会记录一些历史纪录，如此一来，好多了～ 至少很容易知道我修改了哪些数据，以及程序修改的理念与逻辑概念等等，在维护上面是轻松很多很多的喔！

另外，在一些环境的设定上面，毕竟每个人的环境都不相同，为了取得较佳的执行环境， 我都会自行先定义好一些一定会被用到的环境变量，例如 PATH 这个玩意儿！ 这样比较好啦～所以说，建议您一定要养成良好的 script 撰写习惯， 在每个 script 的文件头处记录好：

- script 的功能；
- script 的版本信息；
- script 的作者与联络方式；
- script 的版权宣告方式；
- script 的 History (历史纪录)；
- script 内较特殊的指令，使用绝对路径的方式来下达；
- script 运作时需要的环境变量预先宣告与设定。

简单的 shell script 练习

在第一支 shell script 撰写完毕之后，相信您应该具有基本的撰写功力了。 接下来，在开始更深入的程序概念之前，我们先来玩一些比较有趣的简单的小范例好了。 底下的范例中，达成结果的方式相当的多，

建议您先自行撰写看看，写完之后再与鸟哥写的内容比对， 这样才能更加深概念喔！好！不啰唆，我们就一个一个来玩吧！

- 变量内容由使用者决定

很多时候我们需要使用者输入一些内容，好让程序可以顺利运作。 简单的来说，大家应该都有安装过软件的经验，安装的时候，他不是会问您『要安装到那个目录去？』吗？ 那个让使用者输入的数据的动作，就是让使用者输入变量内容啦。

你应该还记得在 bash 的时候，我们有学到一个 read 指令吧？忘记的话，请自行回头去阅读一番。 现在，请你以 read 指令的用途，撰写一个 script ，他可以让使用者输入：1 first name 与 2. last name， 最后并且在屏幕上显示：『Your full name is: 』的内容：

```
[root@linux scripts]# vi sh02.sh
#!/bin/bash
# Program:
#          Let user keyin their first and last name, and show their full name.
# History:
# 2005/08/23         VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

read -p "Please input your first name: " firstname
read -p "Please input your last name:  " lastname
echo -e "\nYour full name is: $firstname $lastname"
```

将上面这个 sh02.sh 执行一下，你就能够发现使用者自己输入的变量可以被取用的哩！ 很不错吧！加油！

- 利用 date 进行档案的建立

想象一个状况，如果我每天要进行备份，而备份的数据又不想被覆盖掉，也就是说， 我想要将每天备份的数据放在不同的档案中。哇！这真困扰啊？难道要我每天去修改 script ？ 不需要啊！因为每天的『日期』并不相同，所以我可以将档名取成类似： backup.20050802 ， 不就可以每天一个不同档名了吗？呵呵！确实如此。好了，接下来出个例子： 我想要建立三个空的档案，档名最开头由使用者输入决定，假设使用者输入 filename 好了， 那今天的日期是 2005/08/23 ，我想要以前天、昨天、今天的日期来建立这个档案，亦即 filename_20050821, filename_20050822, filename_20050823 ，该如何是好？

```
[root@linux scripts]# vi sh03.sh
#!/bin/bash
# Program:
#          User can keyin filename to touch 3 new files.
# History:
# 2005/08/23         VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH
```

```
# 1. 让使用者输入文件名称，并取得 fileuser 这个变量；
echo -e "I will use 'touch' command to create 3 files."
read -p "Please input the filename what you want: " fileuser

# 2. 为了避免使用者随意按 Enter ，利用变量功能分析文件名是否有设定？
filename=${fileuser:-"filename"}

# 3. 开始利用 date 指令来取得所需要的档名了；
date1=`date --date='2 days ago' +%Y%m%d`
date2=`date --date='1 days ago' +%Y%m%d`
date3=`date +%Y%m%d`
file1="$filename""$date1"
file2="$filename""$date2"
file3="$filename""$date3"

# 4. 将档名建立吧！
touch $file1
touch $file2
touch $file3
```

我透过一些简单的动作，这些动作都可以在 bash 那一章里面找到， 包括小指令 (`) 的取得讯息、变量的设定功能、变量的累加以及利用 touch 指令辅助！ 如果您开始执行这个 sh03.sh 之后，你可以进行两次输入，一次直接按 [Enter] 来查阅档名是啥？ 一次可以输入一些字符，这样来判断你的档案喔！关于 date 的指令应用，请 man date 吧！ ^_^

- 数值运算的方法

各位看官应该还记得，我们可以使用 declare 来定义变量的类型吧？！ 这样才能够进行加减运算啊！可惜的是， bash shell 里头预设仅支持到整数的数据。 OK！那我们来玩玩看，如果我们要使用者输入两个变量，然后将两个变量的内容相乘， 最后输出相乘的结果，那可以怎么做？

```
[root@linux scripts]# vi sh04.sh
#!/bin/bash
# Program:
#         User can input 2 integer to cross by!
# History:
# 2005/08/23        VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH
echo -e "You SHOULD input 2 number, I will cross they! \n"
read -p "first number:  " firstnu
read -p "second number: " secnu
total=$(($firstnu*$secnu))
echo -e "\nThe number $firstnu x $secnu is ==> $total"
```

在数字的运算上，我们可以使用『 declare -i total=$firstnu*$secnu 』 也可以使用上面的方式来进行！基本上，鸟哥比较建议使用这样的方式来进行运算：

var=$((运算内容))

不但容易记忆，而且也比较方便的多～未来您可以使用这种方式来计算的呀！至于数值运算上的处理， 则有：+, -, *, /, %等等。 那个 % 是取余数啦～举例来说， 13 对 3 取余数，结果是 13=4*3+1，所以余数是 1 啊！就是：

```
[root@linux scripts]# nu=$((13%3)); echo $nu
1
```

这样了解了吧？！多多学习与应用喔！ ^_^

善用判断式

在 bash 章节中，我们提到过 $? 这个变量所代表的意义， 此外，也透过 && 及 || 来作为前一个指令是否能够成功进行的一个参考。 那么，如果我想要知道 /dmtsai 这个目录是否存在时，难道一定要使用 ls 来执行， 然后再以 $? 来判断执行成果吗？呵呵！当然不需要！ 我们可以透过『 test 』这个指令来侦测呢！

利用 test 指令的测试功能

当我要检测系统上面某些档案或者是相关的属性时，利用 test 这个指令来工作， 真是好用得不得了，举例来说，我要检查 /dmtsai 是否存在时，使用：

```
[root@linux ~]# test -e /dmtsai
```

执行结果并不会显示任何讯息，但最后我们可以透过 $? 或 && 及 || 来展现整个结果呢！ 例如我们在将上面的例子改写成这样：

```
[root@linux ~]# test -e /dmtsai && echo "exist" || echo "Not exist"
```

最终的结果可以告知我们是『exist』还是『Not exist』呢！那我知道 -e 是测试一个『东西』在不在， 如果还想要测试一下该档名是啥玩意儿时，还有哪些标志可以来判断的呢？呵呵！有底下这些东西喔！

测试的标志	代表意义
1. 关于某个档名的『类型』侦测(存在与否)，如 test -e filename	
-e	该『档名』是否存在？(常用)
-f	该『档名』是否为档案(file)？(常用)
-d	该『文件名』是否为目录(directory)？(常用)
-b	该『文件名』是否为一个 block device 装置？
-c	该『文件名』是否为一个 character device 装置？
-S	该『档名』是否为一个 Socket 档案？
-p	该『档名』是否为一个 FIFO (pipe) 档案？

-L	该『档名』是否为一个连结档?
2. 关于档案的权限侦测，如 test -r filename	
-r	侦测该文件名是否具有『可读』的属性?
-w	侦测该档名是否具有『可写』的属性?
-x	侦测该档名是否具有『可执行』的属性?
-u	侦测该文件名是否具有『SUID』的属性?
-g	侦测该文件名是否具有『SGID』的属性?
-k	侦测该文件名是否具有『Sticky bit』的属性?
-s	侦测该档名是否为『非空白档案』?
3. 两个档案之间的比较，如： test file1 -nt file2	
-nt	(newer than)判断 file1 是否比 file2 新
-ot	(older than)判断 file1 是否比 file2 旧
-ef	判断 file2 与 file2 是否为同一档案，可用在判断 hard link 的判定上。 主要意义在判定，两个档案是否均指向同一个 inode 哩!
4. 关于两个整数之间的判定，例如 test n1 -eq n2	
-eq	两数值相等 (equal)
-ne	两数值不等 (not equal)
-gt	n1 大于 n2 (greater than)
-lt	n1 小于 n2 (less than)
-ge	n1 大于等于 n2 (greater than or equal)
-le	n1 小于等于 n2 (less than or equal)
5. 判定字符串的数据	
test -z string	判定字符串是否为 0 ? 若 string 为空字符串，则为 true
test -n string	判定字符串是否非为 0 ? 若 string 为空字符串，则为 false。 注： -n 亦可省略
test str1 = str2	判定 str1 是否等于 str2 ，若相等，则回传 true
test str1 != str2	判定 str1 是否不等于 str2 ，若相等，则回传 false
6. 多重条件判定，例如： test -r filename -a -x filename	
-a	(and)两状况同时成立!例如 test -r file -a -x file，则 file 同时具有 r 与 x 权限时，才回传 true。
-o	(or)两状况任何一个成立!例如 test -r file -o -x file，则 file 具有 r 或 x 权限时，就可回传 true。
!	反相状态，如 test ! -x file ，当 file 不具有 x 时，回传 true

OK！现在我们就利用 test 来帮我们写几个简单的例子。首先，判断一下， 让使用者输入一个档名，我们判断：

1. 这个档案是否存在，若不存在则给予一个『Filename does not exist』的讯息，并中断程序；
2. 若这个档案存在，则判断他是个档案或目录，结果输出『Filename is regular file』或 『Filename is directory』
3. 判断一下，执行者的身份对这个档案或目录所拥有的权限，并输出权限数据！

你可以先自行创作看看，然后再跟底下的结果讨论讨论。注意利用 test 与 && 还有 || 等标志！

```
[root@linux scripts]# vi sh05.sh
#!/bin/bash
# Program:
#         Let user input a filename, the program will search the filename
#         1.) exist? 2.) file/directory? 3.) file permissions
# History:
# 2005/08/25        VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

# 1. 让使用者输入档名，并且判断使用者是否真的有输入字符串？
echo -e "The program will show you that filename is exist which input by you.\n\n"
read -p "Input a filename : " filename
test -z $filename && echo "You MUST input a filename." && exit 0
# 2. 判断档案是否存在？
test ! -e $filename && echo "The filename $filename DO NOT exist" && exit 0
# 3. 开始判断档案类型与属性
test -f $filename && filetype="regulare file"
test -d $filename && filetype="directory"
test -r $filename && perm="readable"
test -w $filename && perm="$perm writable"
test -x $filename && perm="$perm executable"
# 4. 开始输出信息！
echo "The filename: $filename is a $filetype"
echo "And the permission are : $perm"
```

很有趣的例子吧！您可以自行再以其它的案例来撰写一下可用的功能呢！

利用判断符号 [ ]

除了我们很喜欢使用的 test 之外，其实，我们还可以利用判断符号『 [ ] 』来进行数据的判断呢！ 举例来说，如果我想要知道 $HOME 这个变量是否为空的，可以这样做：

```
[root@linux ~]# [ -z "$HOME" ]
```

但使用 [] 要特别注意的是，在上述的每个组件中间都需要有空格键来分隔，假设我空格键使用『□』来表示， 那么，在这些地方你都需要有空格键：

```
[  "$HOME"  ==  "$MAIL"  ]
[□"$HOME"□==□"$MAIL"□]
 ↑        ↑   ↑        ↑
```

上面的例子在说明，两个字符串 $HOME 与 $MAIL 是否相同的意思，相当于 test $HOME = $MAIL 的意思啦！ 而如果没有空白分隔，例如 [$HOME==$MAIL] 时，我们的 bash 就会显示错误讯息了！这可要很注意啊！ 所以说，您最好要注意：

- 在中括号 [] 内的每个组件都需要有空格键来分隔；
- 在中括号内的变量，最好都以双引号来设定；
- 在中括号内的常数，最好都以单或双引号来设定。

举例来说，假如我设定了 name="VBird Tsai" ，然后这样判定：

```
[root@linux ~]# name="VBird Tsai"
[root@linux ~]# [ $name == "VBird" ]
bash: [: too many arguments
```

为什么呢？因为 $name 如果没有使用双引号刮起来，那么上面的判定式会变成：

[VBird Tsai == "VBird"]

而不是我们要的：

["VBird Tsai" == "VBird"]

这可是差很多的喔！另外，中括号的使用方法与标志与 test 几乎一模一样啊～ 只是中括号比较常用在条件判断式 if ..... then ..... fi 的情况中就是了。 好，那我们也继续来做一个小案例好了：

1. 当执行一个程序的时候，这个程序会让使用者选择 Y 或 N ，
2. 如果使用者输入 Y 或 y 时，就显示『 OK, continue 』
3. 如果使用者输入 n 或 N 时，就显示『 Oh, interrupt ! 』
4. 如果不是 Y/y/N/n 之内的其它字符，就显示『I don't know what is your choise』

利用中括号、 && 与 || 来继续吧！

```
[root@linux scripts]# vi sh06.sh
#!/bin/bash
# Program:
#         This program will show the user's choice
# History:
# 2005/08/25        VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

read -p "Please input (Y/N): " yn
[ "$yn" == "Y" -o "$yn" == "y" ] && echo "OK, continue" && exit 0
[ "$yn" == "N" -o "$yn" == "n" ] && echo "Oh, interrupt!" && exit 0
```

```
echo "I don't know what is your choise" && exit 0
```

很有趣吧！利用这个字符串判别的方法，我们就可以很轻松的将使用者想要进行的工作分门别类呢！ 接下来，我们再来谈一些其它有的没有的东西吧！

> Tips:
> 为什么判断式里面下达等于要用 == 而不是一个 = 就好了呢？我们在前一章正规表示法里面的 awk 提到， 只有一个 = 用来给予一个变量设定其内容，逻辑判断时，则会给予两个等于， 亦即『比较』而非『设定』的意思～这里要好好的分辨一下喔！ ^_^



Shell script 的预设变数($0, $1...)

其实，当我们执行一个 shell script 时，在这个 shell script 里面就已将帮我们做好一些可用的变量了。 举例来说，在不久的将来，您就会发现，当我们要启动一个系统服务时，可能会下达类似这样的指令：

```
[root@linux ~]# /etc/init.d/crond restart
```

那是啥玩意儿？呵呵！就是『向 /etc/init.d/crond 这个 script 下达 restart 的指令』， 咦！我们不是都使用 read 来读取使用者输入的变量内容吗？为啥我可以直接在 script 后面接上这个参数？ 这是因为 shell script 帮我们设定好一些指定的变量了！变量的对应是这样的：

```

/path/to/scriptname  opt1  opt2  opt3  opt4  ...
       $0             $1    $2    $3    $4   ...
```

这样够清楚了吧？！执行的文件名为 $0 这个变量，第一个接的参数就是 $1 啊～ 所以，只要我们在 script 里面善用 $1 的话，就可以很简单的立即下达某些指令功能了！ 好了，来做个例子吧～假设我要执行一个 script ，执行后，该 script 会自动列出自己的档名， 还有后面接的前三个参数，该如何是好？

```
[root@linux scripts]# vi sh07.sh
#!/bin/bash
# Program:
#         The program will show it's name and first 3 parameters.
# History:
# 2005/08/25        VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

echo "The script naem is ==> $0"
[ -n "$1" ] && echo "The 1st paramter is ==> $1" || exit 0
[ -n "$2" ] && echo "The 2nd paramter is ==> $2" || exit 0
[ -n "$3" ] && echo "The 3th paramter is ==> $3" || exit 0
```

这支程序里面鸟哥加上了一些控制式，亦即利用 && 及 || 来加以判断 $1 ~ $3 是否存在？ 若存在才显示，若不存在就中断～执行结果如下：

```
[root@linux scripts]# sh sh07.sh theone haha quot
```

```
The script naem is ==> sh07.sh
The 1st paramter is ==> theone
The 2nd paramter is ==> haha
The 3th paramter is ==> quot
```

上面这 7 个例子都很简单吧？几乎都是利用 bash 的相关功能而已～ 不难啦～底下我们就要使用条件判断式来进行一些分别功能的设定了，好好瞧一瞧先～

条件判断式：

只要讲到『程序』的话，那么条件判断式，亦即是『 if then 』这种判别式肯定一定要学习的！ 因为很多时候，我们都必须要依据某些数据来判断程序该如何进行。举例来说，我们在上头不是有练习当使用者输入 Y/N 时，必须要执行不同的讯息输出吗？简单的方式可以利用 && 与 || ，但如果我还想要执行一堆指令呢？ 那真的得要 if then 来帮忙啰～底下我们就来聊一聊！

利用 if .... then

这个 if .... then 是最常见的条件判断式了～简单的说，就是当符合某个条件判断的时候， 就予以进行某项工作就是了。我们可以简单的这样看：

```
if [ 条件判断式 ]; then
          当条件判断式成立时，可以进行的指令工作内容；
fi
```

至于条件判断式的判断方法，与前一小节的介绍相同啊！较特别的是，如果我有多个条件要判别时， 除了 sh06.sh 那个案例，也就是将多个条件写入一个中括号内的情况之外， 我还可以有多个中括号来隔开喔！ 而括号与括号之间，则以 && 或 || 来隔开，他们的意义是：

- && 代表 AND ；
- || 代表 or ；

所以，在使用中括号的判断式中， && 及 || 就与指令下达的状态不同了。举例来说， sh06.sh 那个例子我可以改写成这样：

```
[root@linux scripts]# vi sh06-2.sh
#!/bin/bash
# Program:
#         This program will show the user's choice
# History:
# 2005/08/25        VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

read -p "Please input (Y/N): " yn
```

```
if [ "$yn" == "Y" ] || [ "$yn" == "y" ]; then
        echo "OK, continue"
        exit 0
fi
if [ "$yn" == "N" ] || [ "$yn" == "n" ]; then
        echo "Oh, interrupt!"
        exit 0
fi
echo "I don't know what is your choise" && exit 0
```

不过，由这个例子看起来，似乎也没有什么了不起吧？ sh06.sh 还比较简单呢～ 但是，如果我们考虑底下的状态，您就会知道 if then 的好处了：

```
if [ 条件判断式 ]; then
        当条件判断式成立时，可以进行的指令工作内容；
else
        当条件判断式不成立时，可以进行的指令工作内容；
fi
```

如果考虑更复杂的情况，则可以使用这个语法：

```
if [ 条件判断式一 ]; then
        当条件判断式一成立时，可以进行的指令工作内容；
elif [ 条件判断式二 ]; then
        当条件判断式二成立时，可以进行的指令工作内容；
else
        当条件判断式一与二均不成立时，可以进行的指令工作内容；
fi
```

那我就可以将 sh06-2.sh 改写成这样：

```
[root@linux scripts]# vi sh06-3.sh
#!/bin/bash
# Program:
#       This program will show the user's choice
# History:
# 2005/08/25        VBird    First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

read -p "Please input (Y/N): " yn

if [ "$yn" == "Y" ] || [ "$yn" == "y" ]; then
        echo "OK, continue"
elif [ "$yn" == "N" ] || [ "$yn" == "n" ]; then
        echo "Oh, interrupt!"
else
```

```
        echo "I don't know what is your choise"
fi
```

是否程序变得很简单，而且依序判断，可以避免掉重复判断的状况，这样真的很容易设计程序的啦！ ^_^ 好了，那么如果我要侦测你所输入的参数是否为 hello 呢 ， 也就是说，如果我想要知道，你在程序后面所接的第一个参数 (就是 $1 啊！) 是否为 hello ，

1. 如果是的话，就显示 "Hello, how are you ?"；
2. 如果没有加任何参数，就提示使用者必须要使用的参数下达法；
3. 而如果加入的参数不是 hello ，就提醒使用者仅能使用 hello 为参数。

整个程序的撰写可以是这样的：

```
[root@linux scripts]# vi sh08.sh
#!/bin/bash
# Program:
#       Show "Hello" from $1....
# History:
# 2005/08/28        VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

if [ "$1" == "hello" ]; then
        echo "Hello, how are you ?"
elif [ "$1" == "" ]; then
        echo "You MUST input parameters, ex> $0 someword"
else
        echo "The only parameter is 'hello'"
fi
```

然后您可以执行这支程序，分别在 $1 的位置输入 hello, 没有输入与随意输入， 就可以看到不同的输出啰～是否还觉得挺简单的啊！ ^_^。事实上， 学到这里，也真的很厉害了～好了，底下我们继续来玩一些比较大一点的啰～ 我们在前一章已经学会了 grep 这个好用的玩意儿，那么多学一个叫做 netstat 的指令， 这个指令可以查询到目前主机有开启的网络服务端口口 (service ports)， 相关的功能我们会在服务器架设篇继续介绍，这里您只要知道，我可以利用『 netstat -tuln 』来取得目前主机有启动的服务，而且取得的信息有点像这样：

```
[root@linux ~]# netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address      Foreign Address      State
tcp        0      0 0.0.0.0:199        0.0.0.0:*            LISTEN
tcp        0      0 :::80              :::*                 LISTEN
tcp        0      0 :::22              :::*                 LISTEN
tcp        0      0 :::25              :::*                 LISTEN
```

上面的重点是特殊字体的那个部分，那些特殊字体的部分代表的就是 port 啰～ 那么每个 port 代表的意义为何呢？几个常见的 port 与相关网络服务的关系是：

- 80: WWW
- 22: ssh
- 21: ftp
- 25: mail

那我如何透过 netstat 去侦测我的主机是否有开启这四个主要的网络服务端口口呢？ 我可以简单的这样去写这个程序喔：

```
[root@linux scripts]# vi sh09.sh
#!/bin/bash
# Program:
#         Using netstat and grep to detect WWW,SSH,FTP and Mail services.
# History:
# 2005/08/28        VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

# 1. 先作一些告知的动作而已～
echo "Now, the services of your Linux system will be detect!"
echo -e "The www, ftp, ssh, and mail will be detect! \n"

# 2. 开始进行一些测试的工作，并且也输出一些信息啰！
testing=`netstat -tuln | grep ":80 "`
if [ "$testing" != "" ]; then
          echo "WWW is running in your system."
fi
testing=`netstat -tuln | grep ":22 "`
if [ "$testing" != "" ]; then
          echo "SSH is running in your system."
fi
testing=`netstat -tuln | grep ":21 "`
if [ "$testing" != "" ]; then
          echo "FTP is running in your system."
fi
testing=`netstat -tuln | grep ":25 "`
if [ "$testing" != "" ]; then
          echo "Mail is running in your system."
fi
```

这样又能够一个一个的检查啰～是否很有趣啊！ ^_^。接下来，我们再来玩更难一点的。 我们知道可以利用 date 来显示日期与时间，也可以利用 $((计算式)) 来计算数值运算。 另外， date 也可以用来显示自 19710101 以来的『总秒数』 (请自行查阅 man date 及 info date) 。那么，您是否可以撰写一支小程序，用来『计算退伍日期还剩几天？』也就是说：

1. 先让使用者输入他们的退伍日期；
2. 再由现在日期比对退伍日期；

3. 由两个日期的比较来显示『还需要几天』才能够退伍的字样。

似乎挺难的样子？其实也不会啦，利用『 date --date="YYYYMMDD" +%s 』就能够达到我们所想要的啰～如果您已经写完了程序，对照底下的写法试看看：

```
[root@linux scripts]# vi sh10.sh
#!/bin/bash
# Program:
#         Tring to calculate your demobilization date at how many days
#         later...
# History:
# 2005/08/29        VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

# 1. 告知使用者这支程序的用途，并且告知应该如何输入日期格式？
echo "This program will try to calculate :"
echo "How many days about your demobilization date..."
read -p "Please input your demobilization date (YYYYMMDD ex>20050401): " date2

# 2. 测试一下，这个输入的内容是否正确？利用正规表示法啰～
date_d=`echo $date2 |grep '[0-9]\{8\}'`
if [ "$date_d" == "" ]; then
          echo "You input the wrong format of date...."
          exit 1
fi

# 3. 开始计算日期啰～
declare -i date_dem=`date --date="$date2" +%s`
declare -i date_now=`date +%s`
declare -i date_total_s=$(($date_dem-$date_now))
declare -i date_d=$(($date_total_s/60/60/24))
if [ "$date_total_s" -lt "0" ]; then
          echo "You had been demobilization before: " $((-1*$date_d)) " ago"
else
          declare -i date_h=$(($(($date_total_s-$date_d*60*60*24))/60/60))
          echo "You will be demobilized after $date_d days and $date_h hours."
fi
```

瞧一瞧，这支程序可以帮您计算退伍日期呢～如果是已经退伍的朋友，还可以知道已经退伍多久了～哈哈！很可爱吧～利用 date 算出自 1971/01/01 以来的总秒数， 再与目前的总秒数来比对，然后以一天的总秒数 (60*60*24) 为基数去计算总日数， 就能够得知两者的差异了～瞧～全部的动作都没有超出我们所学的范围吧～ ^_^ 还能够避免使用者输入错误的数字，所以多了一个正规表示法的判断式呢～ 这个例子比较难，有兴趣想要一探究竟的朋友，可以作一下课后练习题 关于计算生日的那一题喔！～加油！

利用 case ..... esac 判断

上个小节提到的『 if .... then .... fi 』对于变量的判断中， 是以比对的方式来分辨的，如果符合状态就进行某些行为，并且透过较多层次 （ 就是 elif ... ） 的方式来进行多个变量的程序代码撰写，譬如 sh08.sh 那个小程序，就是用这样的方式来的啰。 好，那么万一我有多个既定的变量内容，例如 sh08.sh 当中，我所需要的变量就是 "hello" 及空字符串两个， 那么我只要针对这两个变量来设定状况就好了对吧？！那么可以使用什么方式来设计呢？ 呵呵～就用 case ... in .... esac 吧～，他的语法如下：

```
case $变量名称 in
  "第一个变量内容")
          程序段
          ;;
  "第二个变量内容")
          程序段
          ;;
  *)
          不包含第一个变量内容与第二个变量内容的其它程序执行段
          exit 1
          ;;
esac
```

要注意的是，这个语法是以 case 为开头，而以 esac 为结尾，啥？为何是 esac 呢？想一想，既然 if 的结尾是 fi ，那么 case 的结尾当然就是将 case 倒着写，自然就是 esac 啰～ ^_^，很好记吧～ 另外，每一个变量内容的程序段最后都需要两个分号 (;;) 来代表该程序段落的结束，这挺重要的喔！ 至于为何需要有 * 这个变量内容在最后呢？这是因为，如果使用者不是输入变量内容一或二时， 我们可以告知使用者相关的信息啊！举例来说，我们如果将 sh08.sh 改写的话， 他应该会变成这样喔！

```
[root@linux scripts]# vi sh08-2.sh
#!/bin/bash
# Program:
#          Show "Hello" from $1.... by using case .... esac
# History:
# 2005/08/29          VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

case $1 in
  "hello")
          echo "Hello, how are you ?"
          ;;
  "")
          echo "You MUST input parameters, ex> $0 someword"
          ;;
  *)
          echo "Usage $0 {hello}"
          ;;
```

```
esac
```

在上面这个 sh08-2.sh 的案例当中，如果你输入『 sh sh08-2.sh test 』来执行， 那么屏幕上就会出现『Usage sh08-2.sh {hello}』的字样，告知执行者仅能够使用 hello 喔～ 这样的方式对于需要某些固定字符串来执行的变量内容就显的更加的方便呢？ 这种方式您真的要熟悉喔！这是因为系统的很多服务的启动 scripts 都是使用这种写法的，举例来说，我们 Linux 的服务启动放置目录是在 /etc/init.d/ 当中，我已经知道里头有个 syslog 的服务，我想要重新启动这个服务，可以这样做：

/etc/init.d/syslog restart

重点是那个 restart 啦～如果您进入 /etc/init.d/syslog 就会看到他使用的是 case 语法， 并且会规定某些既定的变量内容，你可以直接下达 /etc/init.d/syslog ， 该 script 就会告知你有哪些后续接的变量可以使用啰～方便吧！ ^_^

一般来说，使用『 case $变量 in 』这个语法中，当中的那个 $变量 大致有两种取得的方式：

- 直接下达式：例如上面提到的，利用『 script.sh variable 』 的方式来直接给予 $1 这个变量的内容，这也是在 /etc/init.d 目录下大多数程序的设计方式。
- 交互式：透过 read 这个指令来让使用者输入变量的内容。

这么说或许您的感受性还不高，好，我们直接写个程序来玩玩：让使用者能够输入 one, two, three ， 并且将使用者的变量显示到屏幕上，如果不是 one, two, three 时，就告知使用者仅有这三种选择。

```
[root@linux scripts]# vi sh11.sh
#!/bin/bash
# Program:
#          Let user input one, two, three and show in screen.
# History:
# 2005/08/29        VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

echo "This program will print your selection !"
# read -p "Input your choice: " choice
# case $choice in
case $1 in
  "one")
          echo "Your choice is ONE"
          ;;
  "two")
          echo "Your choice is TWO"
          ;;
  "three")
          echo "Your choice is THREE"
          ;;
  *)
          echo "Usage {one|two|three}"
```

```
        ;;
esac
```

此时，您可以使用『 sh sh11.sh two 』的方式来下达指令，就可以收到相对应的响应了。 上面使用的是直接下达的方式，而如果使用的是交互式时，那么将上面第 10, 11 行的 "#" 拿掉， 并将 12 行加上批注 (#)，就可以让使用者输入参数啰～这样是否很有趣啊？！

利用 function 功能

什么是『函数 (function)』功能啊？简单的说，其实， 函数可以在 shell script 当中做出一个类似自订执行指令的东西，最大的功能是， 可以简化我们很多的程序代码～举例来说，上面的 sh11.sh 当中，每个输入结果 one, two, three 其实输出的内容都一样啊～那么我就可以使用 function 来简化了！ function 的语法是这样的：

```
function fname() {
        程序段
}
```

那个 fname 就是我们的自订的执行指令名称～而程序段就是我们要他执行的内容了。 要注意的是，在 shell script 当中， function 的设定一定要在程序的最前面， 这样才能够在执行时被找到可用的程序段喔！好～我们将 sh11.sh 改写一下：

```
[root@linux scripts]# vi sh11-2.sh
#!/bin/bash
# Program:
#       Let user input one, two, three and show in screen.
# History:
# 2005/08/29        VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

function printit(){
        echo -n "Your choice is "
}

echo "This program will print your selection !"
case $1 in
  "one")
        printit; echo $1 | tr 'a-z' 'A-Z'
        ;;
  "two")
        printit; echo $1 | tr 'a-z' 'A-Z'
        ;;
  "three")
        printit; echo $1 | tr 'a-z' 'A-Z'
```

```
        ;;
  *)
        echo "Usage {one|two|three}"
        ;;
esac
```

以上面的例子来说，我做了一个函数名称为 printif ，所以，当我在后续的程序段里面， 只要执行 printit 的话， 就表示我的 shell script 要去执行『 function printit .... 』 里面的那几个程序段落啰！ 当然啰，上面这个例子举得太简单了，所以您不会觉得 function 有什么好厉害的， 不过，如果某些程序代码一再地在 script 当中重复时，这个 function 可就重要的多啰～ 不但可以简化程序代码，而且可以做成类似『模块』的玩意儿，真的很棒啦！

另外， function 也是拥有内建变量的～他的内建变量与 shell script 很类似， 函数名称代表示 $0 ，而后续接的变量也是以 $1, $2... 来取代的～ 这里很容易搞错喔～因为『 function fname() { 程序段 } 』内的 $0, $1... 等等与 shell script 的 $0 是不同的。以上面 sh11-2.sh 来说，假如我下达：『 sh sh11-2.sh one 』 这表示在 shell script 内的 $1 为 "one" 这个字符串。但是在 printit() 内的 $1 则与这个 one 无关。 我们将上面的例子再次的改写一下，让您更清楚！

```
[root@linux scripts]# vi sh11-3.sh
#!/bin/bash
# Program:
#       Let user input one, two, three and show in screen.
# History:
# 2005/08/29     VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

function printit(){
        echo "Your choice is $1"
}

echo "This program will print your selection !"
case $1 in
  "one")
        printit 1
        ;;
  "two")
        printit 2
        ;;
  "three")
        printit 3
        ;;
  *)
        echo "Usage {one|two|three}"
        ;;
```

```
esac
```

在上面的例子当中，如果您输入『 sh sh11-3.sh one 』就会出现『 Your choice is 1 』的字样～ 为什么是 1 呢？因为在程序段落当中，我们是写了『 printit 1 』那个 1 就会成为 function 当中的 $1 喔～ 这样是否理解呢？ function 本身其实比较困难一点，如果您还想要进行其它的撰写的话。 不过，我们仅是想要更加了解 shell script 而已，所以，这里看看即可～了解原理就好啰～ ^_^

循环 (loop)

除了 if...then...fi 这种条件判断式之外，循环可能是程序当中最重要的一环了～ 循环可以不断的执行某个程序段落，直到使用者设定的条件达成为止。 所以，重点是那个『条件的达成』是什么。底下我们就来谈一谈：

---

while do done, until do done

一般来说，循环最常见的就是底下这两种状态了：

```
while [ condition ]
do
        程序段落
done
```

这种方式中， while 是『当....时』，所以，这种方式说的是『当 condition 条件成立时，就进行循环，直到 condition 的条件不成立才停止』的意思。

```
until [ condition ]
do
        程序段落
done
```

这种方式恰恰与 while 相反，它说的是『当 condition 条件成立时，就终止循环， 否则就持续进行循环的程序段。』是否刚好相反啊～我们以 while 来做个简单的练习好了。 假设我要让使用者输入 yes 或者是 YES 才结束程序的执行，否则就一直进行告知使用者输入字符串。

```
[root@linux scripts]# vi sh12.sh
#!/bin/bash
# Program:
#       Use loop to try find your input.
# History:
# 2005/08/29        VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

while [ "$yn" != "yes" ] && [ "$yn" != "YES" ]
do
        read -p "Please input yes/YES to stop this program: " yn
```

```
done
```

上面这个例题的说明是『当 $yn 这个变量不是 "yes" 且 $yn 也不是 "YES" 时，才进行循环内的程序。』而如果 $yn 是 "yes" 或 "YES" 时，就会离开循环啰～那如果使用 until 呢？呵呵有趣啰～ 他的条件会变成这样：

```
[root@linux scripts]# vi sh12-2.sh
#!/bin/bash
# Program:
#          Use loop to try find your input.
# History:
# 2005/08/29        VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

until [ "$yn" == "yes" ] || [ "$yn" == "YES" ]
do
          read -p "Please input yes/YES to stop this program: " yn
done
```

仔细比对一下这两个东西有啥不同喔！ ^_^再来，如果我想要计算 1+2+3+....+100 这个数据呢？ 利用循环啊～他是这样的：

```
[root@linux scripts]# vi sh13.sh
#!/bin/bash
# Program:
#          Try to use loop to calculate the result "1+2+3...+100"
# History:
# 2005/08/29        VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

s=0
i=0
while [ "$i" != "100" ]
do
          i=$(($i+1))
          s=$(($s+$i))
done
echo "The result of '1+2+3...+100' is ==> $s"
```

嘿嘿！当您执行了『 sh sh13.sh 』之后，就可以得到 5050 这个数据才对啊！这样瞭呼～ 那么让您自行做一下，如果想要让使用者自行输入一个数字，让程序由 1+2+... 直到您输入的数字为止， 该如何撰写呢？应该很简单吧？！答案可以参考一下习题练习里面的一题喔！

---

for...do....done

相对于 while, until 的循环方式是必须要『符合某个条件』的状态， for 这种语法，则是『 已经知道要进行几次循环』的状态！他的语法是：

```
for (( 初始值; 限制值; 执行步阶 ))
do
          程序段
done
```

这种语法适合于数值方式的运算当中，在 for 后面的括号内的三串内容意义为：

- 初始值：某个变量在循环当中的起始值，直接以类似 i=1 设定好；
- 限制值：当变量的值在这个限制值的范围内，就继续进行循环。例如 i<=100；
- 执行步阶：每作一次循环时，变量的变化量。例如 i=i+1。

值得注意的是，在『执行步阶』的设定上，如果每次增加 1 ，则可以使用类似『i++』的方式，亦即是 i 每次循环都会增加一的意思。好，我们以这种方式来进行 1 累加到 100 的循环吧！

```
[root@linux scripts]# vi sh14.sh
#!/bin/bash
# Program:
#         Try do calculate 1+2+....+100
# History:
# 2005/08/29        VBird     First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

s=0
for (( i=1; i<=100; i=i+1 ))
do
          s=$(($s+$i))
done
echo "The result of '1+2+3+...+100' is ==> $s"
```

一样也是很简单吧！利用这个 for 则可以直接限制循环要进行几次呢！这么好用的东西难道只能在数值方面动作？ 当然不是啦～我们还可以利用底下的方式来进行非数字方面的循环运作喔！

```
for var in con1 con2 con3 ...
do
          程序段
done
```

以上面的例子来说，这个 $var 的变量内容在循环工作时：

1. 第一次循环时， $var 的内容为 con1 ；
2. 第二次循环时， $var 的内容为 con2 ；
3. 第三次循环时， $var 的内容为 con3 ；

4. ....

我们可以做个简单的练习。假设我有三种动物，分别是 dog, cat, elephant 三种， 我想每一行都输出这样：『There are dogs...』之类的字样，则可以：

```
[root@linux scripts]# vi sh15.sh
#!/bin/bash
# Program:
#       Using for .... loop to print 3 animal
# History:
# 2005/08/29      VBird    First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

for animal in dog cat elephant
do
        echo "There are ""$animal""s.... "
done
```

很简单是吧！ ^_^。好了，那么如果我想要让使用者输入某个目录， 然后我找出某目录内的文件名的权限呢？又该如何是好？呵呵！可以这样做啦～

```
[root@linux scripts]# vi sh16.sh
#!/bin/bash
# Program:
#       let user input a directory and find the whole file's permission.
# History:
# 2005/08/29      VBird    First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

# 1. 先看看这个目录是否存在啊？
read -p "Please input a directory: " dir
if [ "$dir" == "" ] || [ ! -d "$dir" ]; then
        echo "The $dir is NOT exist in your system."
        exit 1
fi

# 2. 开始测试档案啰～
filelist=`ls $dir`
for filename in $filelist
do
        perm=""
        test -r "$dir/$filename" && perm="$perm readable"
        test -w "$dir/$filename" && perm="$perm writable"
        test -x "$dir/$filename" && perm="$perm executable"
```

```
        echo "The file $dir/$filename's permission is $perm "
done
```

呵呵！很有趣的例子吧～利用这种方式，您可以很轻易的来处理一些档案的特性呢～ 我们循环就介绍到这里了～其它更多的应用，就得视您的需求来玩啰～。

---

shell script 的追踪与 debug

scripts 在执行之前，最怕的就是出现问题了！那么我们如何 debug 呢？有没有办法不需要透过直接执行该 scripts 就可以来判断是否有问题呢！？呵呵！ 当然是有的！我们就直接以 bash 的相关参数来进行判断吧！

```
[root@linux ~]# sh [-nvx] scripts.sh
参数：
-n  ：不要执行 script，仅查询语法的问题；
-v  ：再执行 sccript 前，先将 scripts 的内容输出到屏幕上；
-x  ：将使用到的 script 内容显示到屏幕上，这是很有用的参数！
范例：

范例一：测试 sh16.sh 有无语法的问题？
[root@linux ~]# sh -n sh16.sh
# 若语法没有问题，则不会显示任何信息！

范例二：将 sh15.sh 的执行过程全部列出来～
[root@linux ~]# sh -x sh15.sh
+ PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/home/vbird/bin
+ export PATH
+ for animal in dog cat elephant
+ echo 'There are dogs.... '
There are dogs....
+ for animal in dog cat elephant
+ echo 'There are cats.... '
There are cats....
+ for animal in dog cat elephant
+ echo 'There are elephants.... '
There are elephants....
# 使用 -x 真的是追踪 script 的好方法，他可以将所有有执行的程序段在执行前列出来，
# 如果是程序段落，则输出时，最前面会加上 + 字号，表示他是程序代码而已，
# 实际的输出则与 standard output 有关啊～如上所示。
```

在上面的范例二当中，我们可以透过这个简单的参数 -x 来达成 debug 的目的，这可是一个不可多得的参数， 通常如果您执行 script 却发生问题时，利用这个 -x 参数，就可以知道问题是发生在哪一行上面了！

熟悉 sh 的用法，将可以使您在管理 Linux 的过程中得心应手！至于在 Shell scripts 的学习方法上面，需要『多看、多模仿、并加以修改成自己的样式！』 是最快的学习手段了！网络上有相当多的朋友在开发一些相当有用的 scripts ，若是您可以将对方的 scripts 拿来，并且改成适合自己主机的样子！那么学

习的效果会是最快的呢！

另外，我们 Linux 系统本来就有很多的启动 script ，如果您想要知道每个 script 所代表的功能是什么？ 可以直接以 vi 进入该 script 去查阅一下，通常立刻就知道该 script 的目的了。 举例来说，我们的 Linux 里头有个文件名称为： /etc/init.d/portmap ，这个 script 是干嘛用的？ 利用 vi 去查阅最前面的几行字，他出现如下信息：

```
# description: The portmapper manages RPC connections, which are used by \
#              protocols such as NFS and NIS. The portmap server must be \
#              running on machines which act as servers for protocols which \
#              make use of the RPC mechanism.
# processname: portmap
```

简单的说，他是被用在 NFS 与 NIS 上面的一个启动 RPC 的 script ， 然后我们再利用 http://www.google.com.tw 去搜寻一下 NFS, NIS 与 RPC ， 立刻就能够知道这个 script 的功能啰～所以，下次您发现不明的 script 时， 如果是系统提供的，那么利用这个检查的方式，一定可以约略了解的啦！ 加油的啰～ ^_^

另外，本章所有的范例都可以在 http://linux.vbird.org/linux_basic/0340bashshell-scripts/scripts.tgz 里头找到喔！加油～

本章习题练习

（ 要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看 ）

- 请建立一支 script ，当你执行该 script 的时候，该 script 可以显示： 1. 你目前的身份 (用 whoami) 2. 你目前所在的目录 (用 pwd)

```
#!/bin/bash
echo -e "Your name is ==> `whoami`"
echo -e "The current directory is ==> `pwd`"
```

- 请自行建立一支程序，该程序可以用来计算『您还有几天可以过生日』啊？？

```
#!/bin/bash
read -p "Pleas input your birthday (MMDD, ex> 0709): " bir
now=`date +%m%d`
if [ "$bir" == "$now" ]; then
echo "Happy Birthday to you!!!"
elif [ "$bir" -gt "$now" ]; then
year=`date +%Y`
total_d=$(($((`date --date="$year$bir" +%s`-`date +%s`))/60/60/24))
echo "Your birthday will be $total_d later"
else
year=$((`date +%Y`+1))
```

```
total_d=$(($((`date --date="$year$bir" +%s`-`date +%s`))/60/60/24))
echo "Your birthday will be $total_d later"
fi
```

- 让使用者输入一个数字，程序可以由 1+2+3... 一直累加到使用者输入的数字为止。

```
#!/bin/bash
read -p "Please input an integer number: " number
i=0
s=0
while [ "$i" != "$number" ]
do
i=$(($i+1))
s=$(($s+$i))
done
echo "the result of '1+2+3+...$number' is ==> $s"
```

- 撰写一支程序，他的作用是: 1.) 先查看一下 /root/test/logical 这个名称是否存在； 2.) 若不存在，则建立一个档案，使用 touch 来建立，建立完成后离开； 3.) 如果存在的话，判断该名称是否为档案，若为档案则将之删除后建立一个档案，档名为 logical ，之后离开； 4.) 如果存在的话，而且该名称为目录，则移除此目录!

```
#!/bin/bash
if [ ! -e logical ]; then
touch logical
echo "Just make a file logical"
exit 1
elif [ -e logical ] && [ -f logical ]; then
rm logical
mkdir logical
echo "remove file ==> logical"
echo "and make directory logical"
exit 1
elif [ -e logical ] && [ -d logical ]; then
rm -rf logical
echo "remove directory ==> logical"
exit 1
else
echo "Does here have anything?"
fi
```

- 我们知道 /etc/passwd 里面以 : 来分隔，第一栏为账号名称。请写一只程序，可以将 /etc/passwd 的第一栏取出，而且每一栏都以一行字符串『The 1 account is "root" 』来显示，那个 1 表示行数。

```
#!/bin/bash
accounts=`cat /etc/passwd | cut -d':' -f1`
for account in $accounts
do
declare -i i=$i+1
echo "The $i account is \"$account\" "
done
```

Linux 账号与身份管理

最近更新日期：2005/09/05

要登入 Linux 系统一定要有账号与密码才行，否则怎么登入，您说是吧？！不过， 不同的使用者应该要拥有不同的权限才行吧？我们还可以透过 user/group 的特殊权限设定， 来规范出不同的群组开发项目呢～在 Linux 的环境下，我们可以透过很多方式来限制使用者能够使用的系统资源， 包括 bash shell 章节提到的 ulimit 限制、还有特殊权限限制，如 umask 等等。 透过这些举动，我们可以规范出不同使用者的使用资源。另外，还记得系统管理员的账号吗？对！ 就是 root 。请问一下，除了 root 之外，是否可以有其它的系统管理员账号？ 为什么大家都要尽量避免使用数字型态的账号？如何修改使用者相关的信息呢？这些我们都得要了解了解的！

1. Linux 的账号与群组
   - 1.1 使用者识别： UID 与 GID
   - 1.2 使用者账号：/etc/passwd, /etc/shadow
   - 1.3 关于群组： 有效与初始群组、 groups, newgrp
2. 账号管理：
   - 2.1 新增与移除使用者： useradd, 相关设定档, passwd, usermod, userdel
   - 2.2 使用者功能：chsh, chfn, finger, id
   - 2.3 新增与移除群组：groupadd, groupmod, groupdel, gpasswd, newgrp
   - 2.4 密码管理： passwd
3. 使用者身份切换：
   - 3.1 su
   - 3.2 sudo, visudo (/etc/sudoers)
4. 使用者的特殊 shell 与 PAM 模块
   - 4.1 特殊的 shell ， /sbin/nologin
   - 4.2 PAM 模块： /etc/nologin, /etc/securetty, /etc/security/*
5. Linux 系统上使用者的对谈与 mail 的使用：
   - 4.1 查询使用者： w, who, last, lastlog
   - 4.2 使用者对谈： talk, mesg, wall
   - 4.3 使用者邮件信箱： mail
6. 手动新增使用者：
   - 5.1 一些检查工具：pwck, pwconv, pwunconv, chpasswd
   - 5.2 特殊账号，如纯数字账号的建立：
   - 5.3 不开放终端机登入的账号 (ex>mail acccount)
   - 5.4 一个大量建置账号的范例：
7. 本章习题练习
8. 针对本文的建议：http://phorum.vbird.org/viewtopic.php?t=23887

Linux 的账号与群组

管理员的工作中，相当重要的一环就是『管理账号』啦！因为整个系统都是你在管理的， 并且所有的一般用户的申请，都必须要透过你的协助才行！所以你就必须要了解一下如何管理好一个网站的账号管理啦！ 在管理 Linux 主机的账号时，我们必须先来了解一下 Linux 到底是如何辨别每一个使用者的！

使用者识别： UID 与 GID

虽然我们登入 Linux 主机的时候，输入的是我们的账号，但是，其实 Linux 主机并不会直接认识你的『账号名称』的，他仅认识 ID 啊～ID 就是一组号码啦～ 主机对于数字比较有概念的，账号只是为了让人们容易记忆而已。 而您的 ID 与账号的对应就在 /etc/passwd 当中哩。

> Tips:
> 如果你曾经以 tarball 安装过软件的话，那么应该不难发现，在解压缩之后的档案， 档案拥有者竟然是『不明的数字』？奇怪吧？ 这没什么好奇怪的，因为 Linux 说实在话，他真的只认识代表你身份的号码而已！



那么到底有几种 ID 呢？还记得我们在『档案属性与目录配置』 那一篇文章的时候有提到每一个档案都具有『拥有人与拥有群组』的属性吗？ 没错啦～每个登入的使用者至少都会取得两个 ID ，一个是使用者 ID (User ID ，简称 UID)、 一个是群组 ID (Group ID ，简称 GID)。

那么档案如何判别他的拥有者与群组呢？其实就是利用 UID 与 GID 啦！ 每一个档案都会有所谓的拥有者 ID 与拥有群组 ID ，亦即是 UID 与 GID ，然后系统会依据 /etc/passwd 的内容，去将该档案的拥有者与群组名称， 使用账号的形式来秀出来！我们可以作个小实验，你可以以 root 的身份 vi /etc/passwd ，然后将你的一般身份的使用者的 ID 随便改一个号码， 然后再到你的一般身份的目录下看看原先该账号拥有的档案，你会发现该档案的拥有人变成了 『数字了』呵呵！这样可以理解了吗？

```
[root@linux ~]# vi /etc/passwd
.....(前面省略).....
dmtsai:x:501:501::/home/dmtsai:/bin/bash  <==将原本的 501:501 改成 3000:501

[root@linux ~]# ls -ld /home/
drwxr-xr-x   3   501 dmtsai 4096 Aug 30 10:37 dmtsai
# 瞧！这里就能够知道，其实档案记录的是 UID 啦～
```

你一定要了解的是，上面的例子仅是在说明 UID 与账号的对应性， 在一部正常运作的 Linux 主机环境下，上面的动作不可随便进行， 这是因为系统上已经有很多的数据在运行了，随意修改系统上某些账号的 UID 很可能会导致某些程序无法进行，这将导致系统无法顺利运作的结果。 因为权限的问题啊！所以，了解了之后，请赶快回到 /etc/passwd 里面，将数字改回来喔！

- 如何登入 Linux 取得 UID/GID

好了，那么我们再来谈一谈，到底我们是怎样登入 Linux 主机的呢？其实也不难啦！当我们在主机前面或者是以 telnet 或者 ssh 登入主机时，系统会出现一个 login 的画面让你输入账号，这个时候当你输入账号与密码之后， Linux 会：

1. 先找寻 /etc/passwd 里面是否有这个账号？如果没有则跳出，如果有的话则将该账号对应的 UID (User ID)与 GID (Group ID)读出来，另外，该账号的家目录与 shell 设定也一并读出；
2. 再来则是核对密码表啦！这时 Linux 会进入 /etc/shadow 里面找出对应的账号与 UID，然后核对一下你刚刚输入的密码与里头的密码是否相符？
3. 如果一切都 OK 的话，就进入 Shell 控管的阶段啰！

大致上的情况就像这样，所以呢，当你要登入你的 Linux 主机的时候，那个 /etc/passwd 与 /etc/shadow 就必须要让系统读取啦，（这也是很多攻击者会将特殊账号写到 /etc/passwd 里头去的缘故！）所以呢，如果你要备份 Linux 的系统的账号的话， 那么这两个档案就一定需要备份才行呦！

使用者账号：/etc/passwd, /etc/shadow

由上面的说明您大概已经知道，嘿嘿！账号管理最重要的两个档案就是『 /etc/passwd 与 /etc/shadow 』了！这两个档案可以说是 Linux 里头最重要的档案之一了！ 如果没有这两个档案的话，呵呵！您可是无法登入 Linux 的呦!所以，底下我们先针对这两个档案来进行说明。当然啰，更详细的数据您可以自行 man 5 passwd 及 man 5 shadow 的啦～

- /etc/passwd

这个档案的构造是这样的：每一行都代表一个账号， 有几行就代表有几个账号在你的系统中！不过需要特别留意的是， 里头很多账号本来就是系统中必须要的，我们可以简称他为系统账号， 例如 bin, daemon, adm, nobody 等等，这些账号是系统正常运作时所需要的，请不要随意的杀掉他呢！ 这个档案的内容有点像这样：

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

我们先来看一下每个 Linux 系统都会有的第一行，就是 root 这个系统管理员那一行好了， 你可以明显的看出来，每一行使用『:』分隔开，共有七个咚咚，分别是：

1. 账号名称：就是账号名称啦！对应 UID 用的！例如 root 就是预设的系统管理员的账号名称；
2. 密码：早期的 Unix 系统的密码是放在这个档案中的， 但是因为这个档案的特性是所有的程序都能够读取，所以，这样一来很容易造成数据的被窃取， 因此后来就将这个字段的密码数据给他改放到 /etc/shadow 中了，关于 /etc/shadow 这一部份等一下再说。而这里你会看到一个 x ，呵呵！别担心，这表示密码已经被移动到 shadow 这个加密过后的档案啰；
3. UID：这个就是使用者识别码 (ID) 啰！通常 Linux 对于 UID 有几个限制需要说给您了解一下：

id 范围	该 ID 使用者特性
0	当 UID 是 0 时，代表这个账号是『系统管理员』！所以当你要作另一个系统管理员账号时， 你可以将该账号的 UID 改成 0 即可；这也就是说，一部系统上面的系统管理员不见得只有 root 喔！ 不过，不很建议有多个账号的 UID 是 0 啦～
1~499	保留给系统使用的 ID，其实 1~65534 之间的账号并没有不同， 也就是除了 0 之外，其它的 UID 并没有不一样，预设 500 以下给系统作为保留账号只是一个习惯。这样的好处是，以有名的 DNS 服务器的启动服务『 named 』为例，这个程序的预设所有人 named 的账号 UID 是 25 ，当有其它的账号同样是 25 时，很可能会造成系统的一些小问题！为了杜绝这样的问题，建议保留 500 以前的 UID 给系统吧！ 不过，一般来说， 1~99 会保留给系统预设的账号，另外 100~499 则保留给一

	些服务来使用。
500~65535	给一般使用者用的。事实上，目前的 linux 核心 (2.6.x 版)已经可以支持到 4294967295 (2^32-1) 这么大的 UID 号码喔！

4.
   上面这样说明可以了解了吗？是的， UID 为 0 的时候，就是 root 呦！所以请特别留意一下你的 /etc/passwd 档案！

5. GID：这个与 /etc/group 有关！其实 /etc/group 的观念与 /etc/passwd 差不多，只是他是用来规范 group 的而已！
6. 使用者信息说明栏：这个字段基本上并没有什么重要用途， 只是用来解释这个账号的意义而已！不过，如果您提供使用 finger 的功能时， 这个字段可以提供很多的讯息呢！底下的 chfn 可以解释一下啰！
7. 家目录：这是使用者的家目录，以上面为例， root 的家目录在 /root ，所以当 root 登入之后，就会立刻跑到 /root 里头啦！ 呵呵！ 如果你有个账号的使用空间特别的大，你想要将该账号的家目录移动到其它的硬盘去， 没有错！可以在这里进行修改呦！预设的使用者家目录在 /home/yourIDname
8. Shell：所谓的 shell 是用来沟通人类下达的指令与硬件之间真正动作的界面！我们通常使用 /bin/bash 这个 shell 来进行指令的下达!嘿嘿!发现了吧？我们在 bash 章节里面提到很多次，登入 Linux 时为何预设是 bash 呢？就是这里设定的啦～ 这里比较需要注意的是，有一个 shell 可以用来替代成让账号无法登入的指令！那就是 /sbin/nologin 这个东西！这也可以用来制作纯 pop 邮件账号者的数据呢！

- /etc/shadow

上面约略提到，由于每个程序都需要取得 uid 与 gid 来判断权限的问题，所以， /etc/passwd 的权限必须要设定成为 -rw-r--r-- 这样的权限，在这样的情况下， 使用者的密码不就任何人都可以看到吗？即使这个档案内的密码栏是加密的， 坏心肠的朋友也可能利用暴力破解法去 try and error 找出您的密码数据.....

因为这样的关系，所以后来发展出将密码移动到 /etc/shadow 这个档案分隔开来的技术， 而且还加入很多的密码限制参数在 /etc/shadow 里头呢！我们先来了解一下这个档案的构造吧！ 我的 /etc/shadow 档案有点像这样：

```
root:$1$i9Ejldjfjio389u9sjl$jljsoi45QE/:12959:0:99999:7:::
bin:*:12959:0:99999:7:::
daemon:*:12959:0:99999:7:::
adm:*:12959:0:99999:7:::
```

基本上， shadow 同样以『:』作为分隔符，如果数一数，会发现共有九个字段啊， 这九个字段的用途是这样的：

1. 账号名称：由于密码也需要与账号对应啊～因此， 这个档案的第一栏就是账号，必须要与 /etc/passwd 相同才行！

2. 密码：这个才是真正的密码，而且是 经过编码过的密码啦！ 你只会看到有一些特殊符号的字母就是了！需要特别留意的是， 虽然这些加密过的密码很难被解出来，但是『很难』不等于『不会』，所以， 这个档案的预设属性是『-rw-------』或者是『-r--------』，亦即只有 root 才可以读写就是了！你得随时注意，不要不小心更动了这个档案的属性呢！另外， 如果是在密码栏的第一个字符为『 * 』或者是『 ! 』，表示这个账号并不会被用来登入的意思。 所以万一哪一天你的某个使用者不乖时，可以先在这个档案中，将他的密码字段的最前面多加一个 * ！嘿嘿！他就无法使用该账号啰！直到他变乖了，再给他启用啊！
3. 最近更动密码的日期：这个字段记录了『更动密码的那一天』的日期， 不过，很奇怪呀！在我的例子中怎么会是 12959 呢？呵呵，这个是因为计算 Linux 日期的时间是以 1970 年 1 月 1 日作为 1 ，而 1971 年 1 月 1 日则为 366 啦！ 所以这个日期是累加的呢！得注意一下这个资料呦！那么最近的 2005 年 1 月 1 日就是 12784 啦，了解了吗？
4. 密码不可被更动的天数： 第四个字段记录了这个账号的密码需要经过几天才可以被变更！如果是 0 的话， 表示密码随时可以更动的意思。这的限制是为了怕密码被某些人一改再改而设计的！如果设定为 20 天的话，那么当你设定了密码之后， 20 天之内都无法改变这个密码呦！
5. 密码需要重新变更的天数： 由于害怕密码被某些『有心人士』窃取而危害到整个系统的安全，所以有了这个字段的设计。 你必须要在这个时间之内重新设定你的密码，否则这个账号将会暂时失效。 而如果像上面的 99999 的话，那就表示，呵呵，密码不需要重新输入啦！ 不过，如果是为了安全性，最好可以设定一段时间之后，严格要求使用者变更密码呢！
6. 密码需要变更期限前的警告期限：当账号的密码失效期限快要到的时候， 就是上面那个『必须变更密码』的那个时间时， 系统会依据这个字段的设定，发出『警告』言论给这个账号，提醒他『再过 n 天你的密码就要失效了，请尽快重新设定你的密码呦！』，如上面的例子，则是密码到期之前的 7 天之内，系统会警告该用户。
7. 密码过期的恕限时间：如果用户过了警告期限没有重新输入密码， 使得密码失效了，也就是说，你在『必须变更密码的期限前，并没有变更你的密码！』 那么该组密码就称为『失效的密码』啰～怎么办？没关系，还有这个字段的天数设计啊～ 意思就是说，当密码失效后，你还可以用这个密码在 n 天内进行登入的意思。 而如果在这个天数后还是没有变更密码，呵呵！那么您的账号就失效了！无法登入！
8. 账号失效日期：这个日期跟第三个字段一样，都是使用 1970 年以来的总日数设定。这个字段表示： 这个账号在此字段规定的日期之后，将无法再使用。 这个字段会被使用通常应该是在『收费服务』的系统中， 你可以规定一个日期让该账号不能再使用啦！
9. 保留：最后一个字段是保留的，看以后有没有新功能加入。

举个例子来说好了，假如我的 dmtsai 这个使用者的密码栏如下所示：

```
dmtsai:$1$8zdAKdfC$XDa8eSus2I7nQL7UjRsIy/:13025:5:60:7:2:13125:
```

这表示什么呢？要注意的是， 13025 是 2005/08/30 ，所以， dmtsai 这个使用者他的密码相关意义是：

- 最近一次更动密码的日期是 2005/08/30 (13025)；
- 能够修改密码的时间是 5 天以后，也就是 2005/09/04 以前 dmtsai 不能修改自己的密码； 如果使用者还是尝试要更动自己的密码，系统就会出现这样的讯息：

```
You must wait longer to change your password
passwd: Authentication token manipulation error
```

- 使用者必须要在 2005/09/04 到 2005/10/29 之间的 60 天限制内去修改自己的密码，若 2005/10/29 之后还是没有变更密码时，该账号就会宣告失效；
- 如果使用者一直没有更改密码，那么在 2005/10/29 之前的 7 天内，系统会警告 dmtsai 应该修改密码的相关信息；例如当 dmtsai 登入时，系统会主动提示如下的信息：

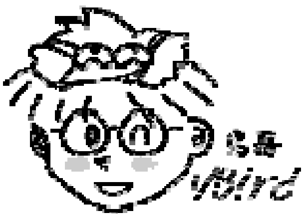
```
Warning: your password will expire in 5 days
```

- 如果该账号一直到 2005/10/29 都没有更改密码，由于还有两天的恕限时间，因此， dmtsai 还是可以在 2005/10/31 以前继续登入；
- 如果使用者在 2005/10/29 以前变更过密码，那么那个 13025 的日期就会跟着改变，因此， 所有的限制日期也会跟着相对变动喔！^_^
- 无论使用者如何动作，到了 13125 ，大约是 2005/12/8 左右，该账号就失效了～

透过这样的说明，您应该会比较容易理解了吧？！ ^_^

Tips:
常常听到：我的密码忘记或者被更动了？怎么办？

有的时候会发生这样的情况，就是说，你的 root 密码忘记了！要怎么办？重新安装吗？另外， 有的时候是被入侵了， root 的密码被更动过，该如何是好？



这个时候就必须要使用到 /etc/shadow 这个数据了！我们刚刚知道密码是存在这个档案中的， 所以只要你能够以各种可行的方法开机进入 Linux ，例如单人维护模式，或者是以 live CD (KNOPPIX) 来进入 Linux 系统。之后，将硬盘顺利挂载，然后进入 /etc/shadow 这个档案中，将 root 的密码这一栏全部清空！然后再登入 Linux 一次，这个时候 root 将不需要密码 (有的时候需要输入空格符) 就可以登入了!这个时候请赶快以 passwd 设定 root 密码即可。

关于群组： 有效与初始群组、groups, newgrp

认识了账号相关的两个档案 /etc/passwd 与 /etc/shadow 之后，您或许还是会觉得奇怪， 那么群组的设定档在哪里？还有，在 /etc/passwd 的第四栏不是所谓的 GID 吗？那又是啥？ 呵呵～此时就需要了解 /etc/group 与 /etc/gshadow 啰～

- /etc/group

这个档案就是在记录 GID 与群组名称的对应了～我的 /etc/group 内容有点像这样：

```
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
```

也是以冒号『:』作为字段的分隔符，共分为四栏，每一字段的意义是：

1. 群组名称：就是群组名称啦！
2. 群组密码：通常不需要设定，因为我们很少使用到群组登入！ 不过，同样的，密码也是被纪录在 /etc/gshadow 当中啰！
3. GID：就是群组的 ID 啊～
4. 支持的账号名称：加入这个群组里面的所有的账号， 我们知道，一个使用者是可以加入多个群组的。举例来说，如果我想要让 dmtsai 也加入 root 这个群组，那么在第一行的最后面加上『,dmtsai』，注意不要有空格， 使成为『 root:x:0:root,dmtsai』就可以啰～

比较重要的特色在于第四栏啦，因为每个使用者都可以拥有多个支持的群组，这就好比在学校念书的时候，我们可以加入多个社团一样！ ^_^。 不过这里您或许会觉得奇怪的，那就是：『假如我同时加入多个群组，那么我在作业的时候，到底是以那个群组为准？』底下我们就来谈一谈这个『有效群组』的概念。

- 有效群组(effective group)与初始群组(initial group)

还记得每个使用者在他的 /etc/passwd 里面的第四栏有所谓的 GID 吧？那个 GID 就是所谓的『初始群组 (initial group) 』了！也就是说，当使用者一登入系统，立刻就拥有这个群组的相关权限的意思。 举例来说，我们上面提到 dmtsai 这个使用者的 /etc/passwd 与 /etc/group 还有 /etc/gshadow 相关的内容如下：

```
[root@linux ~]# grep dmtsai /etc/passwd /etc/group /etc/gshadow
/etc/passwd:dmtsai:x:501:501::/home/dmtsai:/bin/bash
/etc/group:users:x:100:dmtsai
/etc/group:dmtsai:x:501:
/etc/gshadow:users:::dmtsai
/etc/gshadow:dmtsai:!::
```

仔细看到上面这个表格，在 /etc/passwd 里面，dmtsai 这个使用者所属的群组为 GID=501 ， 也就是 /etc/group 里头 dmtsai 那个群组啦～因为这是 initial group ，所以， 使用者一登入就会主动取得，不需要在 /etc/group 的第四个字段写入该账号的！

但是非 initial group 的其它群组可就不同了。举上面这个例子来说，我将 dmtsai 加入 users 这个群组当中，由于 users 这个群组并非是 dmtsai 的初始群组，因此， 我必须要在 /etc/group 这个档案中，找到 users 那一行，并且将 dmtsai 这个账号加入第四栏，这样 dmtsai 才能够支持 users 这个群组啊。

那么在这个例子当中，因为我的 dmtsai 这个账号同时支持 dmtsai 与 users 这两个群组， 因此，在读取/写入/执行档案时，针对群组部分，只要是 users 与 dmtsai 这两个群组拥有的功能， 我 dmtsai 这个使用者都能够拥有喔！这样瞭呼？不过，这是针对已经存在的档案而言， 如果今天我要建立一个新的档案或者是新的目录，请问一下，新档案的群组是 dmtsai 还是 users ？ 呵呵！这就得要检查一下当时的有效群组了 (effective group)。

如果我以 dmtsai 这个使用者的身份登入后，该如何知道我所有支持的群组呢？ 很简单啊，直接输入 groups 就可以了！注意喔，是 groups 有加 s 呢！结果像这样：

```
[dmtsai@linux ~]$ groups
```

```
dmtsai users
```

在这个输出的讯息中，我知道我同时属于 dmtsai 及 users 这个两个群组，而且， 第一个输出的群组即为有效群组 (effective group) 了。 也就是说，我的有效群组为 dmtsai 啦～此时，如果我以 touch 去建立一个新档，例如： touch test ，那么这个档案的拥有者为 dmtsai ，而且群组也是 dmtsai 的啦。 这样是否可以了解什么是有效群组了？

那么如何变更有效群组呢？这个有两个方法，不论是那个方法，都是以 newgrp 达成的！ 以上面这个例子来说，因为我的 dmtsai 使用者同时拥有 dmtsai 与 users 两个群组，因此， dmtsai 当然可以随时切换 dmtsai/users 成为有效群组啰。所以，我可以下达：

```
[dmtsai@linux ~]$ newgrp users
[dmtsai@linux ~]$ groups
users dmtsai
```

此时，我的有效群组就成为 users 了。当然，要能够顺利切换有效群组的话，还需要 /etc/gshadow 的辅助才行～这个等一下我们会说明的。好了，那么如果你开始在 /home/dmtsai 这个家目录底下尝试建立一个档案，例如『 touch test2 』好了，会发生什么状态呢？ 呵呵！那个档案的群组竟然变成 users 了！这样更清楚有效群组的意义了吧？！

我们额外的来讨论一下 newgrp 这个指令，这个指令可以变更目前使用者的有效群组， 而且是另外以一个 shell 来提供登入的喔，所以，以上面的例子来说， dmtsai 这个使用者目前是以另一个 shell 登入的，而且新的 shell 给予 dmtsai 有效 GID 为 users 就是了。当直接执行『 newgrp groupname 』时，使用者的有效群组会成为 groupname ， 此时虽然使用者的环境设定(例如环境变量等等其它数据)不会有影响，但是使用者的『权限』将会重新被计算。 举例来说， dmtsai 此时建立的新档案群组是 users 了～

鸟哥的这个例子当中，要注意的是， dmtsai 这个使用者本来就属于 users 与 dmtsai 这两个群组， 所以他可以直接使用 newgrp 来切换有效群组，而要离开新的有效群组时，输入『 exit 』即可。 假设我的 Linux 系统当中还有另一个群组，名称为 vbird，那么 dmtsai 是否可以登入 vbird 这个群组？ 在某些前提下是可以的：

- vbird 这个群组在 /etc/gshadow 的密码栏为合法的(不具有 ！ 开头！)；
- dmtsai 必须让 root 或群组管理员 (group administrator) 加入到 vbird 群组中。

这两个大前提缺一不可喔!好了,假设我已经使用 gpasswd 建立了 vbird 这个群组的密码， 而 dmtsai 也被加入群组成员当中了，那么当 dmtsai 输入 『newgrp vbird』时， 嘿嘿！ dmtsai 这个使用者的有效群组就能够变成 vbird 啰～

- /etc/gshadow

刚刚讲了很多关于『有效群组』的概念，另外，也提到 newgrp 这个指令的用法， 但是，如果 /etc/gshadow 这个设定没有搞懂得话，那么 newgrp 是无法动作的呢！ 我的 /etc/gshadow 的内容有点像这样：

```
root:::root
bin:::root,bin,daemon
daemon:::root,bin,daemon
sys:::root,bin,adm
```

同样还是使用冒号『:』来作为字段的分隔字符，而且你会发现，这个档案几乎与 /etc/group 一模一样啊！是这样没错～不过，要注意的大概就是第二个字段吧～第二个字段是密码栏， 如果密码栏上面是『!』时，表示该群组不能使用密码来登入呢！ 至于第四个字段也就是支持的账号名称啰～

1. 群组名称
2. 密码栏，同样的，开头为 ! 表示无法登入；
3. 群组管理员的账号 (相关信息在后续介绍)
4. 该群组的所属账号 (与 /etc/group 内容相同！)

不过，就以系统的操作来说，事实上，这个 /etc/gshadow 的密码提供，最大的功能是在于『 让那些不在群组中的成员，临时加入该群组用的。 』 实际上使用的情况是很少的～而如果真的要操作这样的环境，那就得要熟悉 newgrp 的用法啰！ 而且还要提供某个群组的密码出来，真是不好管理。所以，若真的想要让某个使用者利用该群组的功能时， 还是直接将对方加入群组的支持就好了！省得麻烦～

账号管理

好啦！既然要管理账号，当然是由新增与移除使用者开始的啰～底下我们就分别来谈一谈如何新增、 移除与更改使用者的相关信息吧～

新增与移除使用者： useradd, 相关设定档, passwd, usermod, userdel

要如何在 Linux 的系统新增一个使用者啊？呵呵～真是太简单了～直接利用 useradd 这个指令即可！ 他的指令下达方法如下：

- useradd

```
[root@linux ~]# useradd [-u UID] [-g initial_group] [-G other_group] \
>  -[Mm] [-c 说明栏] [-d home] [-s shell] username
参数：
-u  ：后面接的是 UID ，是一组数字。直接指定一个特定的 UID 给这个账号；
-g  ：后面接的那个群组名称就是我们上面提到的 initial group 啦～
      该 group ID (GID) 会被放置到 /etc/passwd 的第四个字段内。
-G  ：后面接的群组名称则是这个账号还可以支持的群组。
      这个参数会修改 /etc/group 内的相关资料喔！
-M  ：强制！不要建立使用者家目录
-m  ：强制！要建立使用者家目录！
-c  ：这个就是 /etc/passwd 的第五栏的说明内容啦～可以随便我们设定的啦～
-d  ：指定某个目录成为家目录，而不要使用默认值；
-r  ：建立一个系统的账号，这个账号的 UID 会有限制 (/etc/login.defs)
-s  ：后面接一个 shell ，预设是 /bin/bash 的啦～
范例：

范例一：完全参考默认值建立一个使用者，名称为 vbird1
```

```
[root@linux ~]# useradd vbird1
[root@linux ~]# ls -l /home
drwxr-xr-x   3 vbird1 vbird1 4096 Aug 30 17:33 vbird1
[root@linux ~]# grep vbird1 /etc/passwd /etc/shadow /etc/group
/etc/passwd:vbird1:x:502:502::/home/vbird1:/bin/bash
/etc/shadow:vbird1:!!:13025:0:99999:7:::
/etc/group:vbird1:x:502:
# 做这个范例只是想要让您了解，其实系统已经规范好了一些新增使用者时的参数了！
# 因此，当我们使用 useradd 时，系统会主动的去修改 /etc/passwd 与 /etc/shadow，
# 而这两个档案内的相关字段参考值，则会以一些设定档的内容来规范喔！
# 同时也要注意到，使用 useradd 新增使用者时，这个使用者的 /etc/shadow
# 密码栏会是不可登入的 (以 !! 为开头)，因此还需要使用 passwd
# 来给予 vbird1 密码后，才算新增完毕！

范例二：我知道我的系统当中有个群组名称为 users ， 且 UID 700 并不存在，
        请用这两个参数给予 vbird2 建立一个账号！
[root@linux ~]# useradd -u 700 -g users vbird2
[root@linux ~]# ls -l /home
drwxr-xr-x   3 vbird2 users  4096 Aug 30 17:43 vbird2
[root@linux ~]# grep vbird2 /etc/passwd /etc/shadow /etc/group
/etc/passwd:vbird2:x:700:100::/home/vbird2:/bin/bash
/etc/shadow:vbird2:!!:13025:0:99999:7:::
# 看一下，UID 与 initial group 确实改变成我们需要的了！

范例三：建立一个系统账号，名称为 vbird3
[root@linux ~]# usradd -r vbird3
[root@linux ~]# grep vbird3 /etc/passwd /etc/shadow /etc/group
/etc/passwd:vbird3:x:101:102::/home/vbird3:/bin/bash
/etc/shadow:vbird3:!!:13025::::::
/etc/group:vbird3:x:102:
# 很重要喔！您会发现， UID 竟然是 101 ，而 GID 怎么会是 102，
# 并且与 /etc/group 有对应的关系喔！有没有加 -r 差很多ㄟ！
```

我的天呐！这个指令更动的档案怎么怎么多啊？对啊！你才知道啊～ 这也是为啥我们说账号管理是很复杂的啦～而且他参考的设定档才更多哩！ 这个指令至少可能会更动到的地方有：

- /etc/passwd
- /etc/shadow
- /etc/group
- /etc/gshadow
- /home/username

那请教一下，您有没有想过，以上述的第一个范例一好了，为何 useradd vbird1 他会主动在 /home/vbird1 建立起使用者的家目录？家目录内有什么数据且来自哪里？ 为何预设使用的是 /bin/bash 这个 shell ？呵呵！这就得要说明一下 useradd 所使用的参考档案啰！

- 相关设定档

我们使用 useradd 去新增使用者时，一些在 /etc/passwd 当中的值会去参考『 /etc/default/useradd 』，这个档案的内容有点像这样：

```
GROUP=100            <==预设的群组
HOME=/home           <==预设的家目录所在目录
INACTIVE=-1                    <==在 /etc/shadow 内的第 7 栏
EXPIRE=                        <==在 /etc/shadow 内的第 8 栏
SHELL=/bin/bash                <==预设的 shell
SKEL=/etc/skel                 <==使用者家目录的内容数据参考目录
```

关于群组的建立机制：

当我们直接使用 useradd 来新增账号时，在预设的情况下，相关的信息都是参考 /etc/default/useradd 这个档案内容的设定的。不过，对于使用者群组的建立机制中， 则有两种不一样的机制存在的：

- 以 FC4 为代表，新建使用者时，若无指定 initial group ， 则系统会主动建立一个与账号相同的群组名称，以该群组作为使用者的 initial group;
- 以 SuSE 9 为代表，新建使用者时，预设不会建立新群组，而以 /etc/default/useradd 内的 GROUP 设定值作为使用者的 initial group 。

这应该是很容易理解才是～如果看不懂，请回去前一小节查阅一下 /etc/passwd, /etc/shadow 的相关内容架构。

关于使用者家目录的参考：/etc/skel/*

在这个档案当中，比较奇怪的是 SKEL 这个玩意儿了，他是啥？ 其实，这个咚咚就是使用者家目录的参考目录啰～举我们的范例一为例，我利用 useradd vbird1 时，他在 /home/vbird1 这个使用者家目录内的各项数据，都是由 /etc/skel 所复制过去的～所以呢，未来如果我想要让新增使用者时，该使用者的环境变量 ~/.bashrc 就设定妥当的话，您可以到 /etc/skel/.bashrc 去编辑一下，也可以建立 /etc/skel/public_html 这个目录，那么未来新增使用者后，在他的家目录下就会有 public_html 那个目录了！这样瞭呼？

关于使用者 UID/GID 的设定：

另外，与密码还有 UID/GID 有关的设定档则是在 /etc/login.defs 里面， 这个档案有点像这样：

```
MAIL_DIR        /var/spool/mail          <==使用者预设邮件信箱放置目录

PASS_MAX_DAYS   99999                    <==/etc/shadow 内的第 5 栏
PASS_MIN_DAYS   0              <==/etc/shadow 内的第 4 栏
PASS_MIN_LEN    5              <==密码最短的字符长度，建议可以改到 6 以上
PASS_WARN_AGE   7              <==/etc/shadow 内的第 6 栏
```

```
UID_MIN          500 <==使用者最小的 UID，意即小于 500 的 UID 为系统保留
UID_MAX        60000 <==使用者能够用的最大 UID
GID_MIN          500 <==使用者自订群组的最小 GID，小于 500 为系统保留
GID_MAX        60000 <==使用者自订群组的最大 GID

CREATE_HOME      yes <==在不加 -M 及 -m 时，是否主动建立使用者家目录？
```

看到这个档案后，您应该晓得的是，为何新建的使用者的 UID 都会大于 500 了吧？ 而且某些版本的 distributions (例如 SuSE server 9) 则是将 UID_MIN 设定为 1000 ， 所以，他的一般身份使用者的 UID 就会从 1000 起跳啰～这样了解吗？！

那如果我现在新增一个使用者，这个使用者的 UID 会是多少？答案是：『 如果 /etc/passwd 里面的账号所属的 UID 没有大于 /etc/login.defs 里头的 UID_MIN (在本例中是 500)时，则以 UID 500 来作为一个新账号的 UID。 如果 /etc/passwd 已有大于 500 以上的 UID 时，则取 /etc/passwd 内最大的那个 UID + 1 作为新设帐号的 UID。』而如果我是想要建立系统用的账号，所以使用 useradd -r sysaccount 这个 -r 的参数时，就会找『比 500 小的最大的那个 UID + 1 』就是了。 ^_^

关于家目录预设是否建立：

另外也要注意那个 CREATE_HOME 的设定值，这个设定值也很重要。一般来说，在 FC4 的环境下， 我们使用 useradd useraccount 时，预设是会主动的建立家目录的，除非使用 -M 这个参数～ 至于 SuSE server 9 这个版本来说，嘿嘿！他预设是不建立家目录的，除非使用 -m 这个参数呢！ 因此，在这里鸟哥也要建议您，如果肯定要建立家目录的话，不论在那个版本， 你最好还是加上 -m 这个参数来强制建立家目录吧！ ^_^

那么您就能知道啰， useradd 这支程序在建立 Linux 上的账号时，至少会参考：

- /etc/default/useradd
- /etc/login.defs
- /etc/skel/*

这些档案，不过，最重要的其实是建立 /etc/passwd, /etc/shadow, /etc/group, /etc/gshadow 还有使用者家目录就是了～所以，如果您了解整个系统运作的状态， 也是可以手动直接修改这几个档案就是了。

- passwd

刚刚我们讲到了，使用 useradd 建立了账号之后，在预设的情况下，该账号是暂时被封锁的， 也就是说，该账号是无法登入的，你可以去瞧一瞧 /etc/shadow 内的第二个字段就晓得啰～ 那该如何是好？怕什么？直接给他设定新密码就好了嘛！对吧～设定密码就使用 passwd 啰！

```
[root@linux ~]# passwd [useraccount]

范例一：如果 root 要帮 dmtsai 修改密码时？
[root@linux ~]# passwd dmtsai
Changing password for user dmtsai.
```

```
New UNIX password:  <==这里直接输入新的密码，屏幕不会有任何反应
BAD PASSWORD: it is based on a dictionary word  <==密码太简单时的错误！
Retype new UNIX password:  <==再输入一次同样的密码
passwd: all authentication tokens updated successfully.  <==竟然还是成功修改了！

范例二： dmtsai 这个使用者想要修改自己的密码时
[dmtsai@linux ~]$ passwd
Changing password for user dmtsai.
Changing password for dmtsai
(current) UNIX password: <==这里输入『原有的旧密码』
New password: <==这里输入新密码
BAD PASSWORD: it is based on your username <==密码的规范是很严格的
New password:
BAD PASSWORD: it is based on your username
New password:
BAD PASSWORD: it is based on a dictionary word
passwd: Authentication token manipulation error
```

先来谈一谈上面的两个范例。要注意的是， passwd 这个指令由于使用者的身份而有两种用法， 如果是 root ，由于 root 具有至高无上的权力，所以 root 可以利用 passwd [username] 来帮使用者修改他们的密码！因此，『如果使用者的密码不见了， root 是可以帮他们进行密码的修改，而不需要知道旧密码。』另外，也只有 root 可以随便设定密码，即使该密码并不符合系统的密码验证要求～ @_@。例如上面的范例一， 我帮 dmtsai 建立的密码太简单，所以其实系统是『警告』过 root 的。 但在重复输入两次密码后，嘿嘿！您还是会看到 successfully 这个成功的字样呢！

那么如果是一般身份使用者，或者是 root 想要修改自己的密码时，直接输入『 passwd 』， 就能够修改自己的密码了。一般身份使用者输入的密码会经过系统的验证， 验证的机制除了 /etc/login.defs 里头规定的最小密码字符数之外，还会受到 /etc/pam.d/passwd 这个 PAM 模块的检验呢！一般来说，您输入的密码最好要符合底下的要求：

- 密码不能与账号相同；
- 密码尽量不要选用字典里面会出现的字符串；
- 密码需要超过 8 个字符；

如果无法经过验证，那么该密码就不被接受，当然还是只能使用旧密码啰！ 此外，仅能接受三次密码输入，如果输入的密码都不被接受，那只好....重新执行一次 passwd 啊！而， 经过这个 passwd [username] 的动作后，您的账号就会有密码啰，此时， 如果察看一下 /etc/shadow ，你就会知道密码内容被改过啰～ ^_^

- usermod

所谓这『人有失手，马有乱蹄』，您说是吧？所以啰，当然有的时候会『不小心』在 useradd 的时候加入了错误的设定数据。或者是，在使用 useradd 后，发现某些地方还可以进行细部修改。 此时，当然我们可以直接到 /etc/passwd 或 /etc/shadow 去修改相对应字段的数据， 不过，Linux 也有提供相关的指令让大家来进行账号相关数据的微调呢～那就是 usermod 啰～

```
[root@linux ~]# usermod [-cdegGlsuLU] username
参数：
-c  ：后面接账号的说明，即 /etc/passwd 第五栏的说明栏，可以加入一些账号的说明。
-d  ：后面接账号的家目录，即修改 /etc/passwd 的第六栏；
-e  ：后面接日期，格式是 YYYY-MM-DD 也就是在 /etc/shadow 内的第八个字段数据啦！
-g  ：后面接 group name，修改 /etc/passwd 的第四个字段，亦即是 GID 的字段！
-G  ：后面接 group name，修改这个使用者能够支持的群组，修改的是 /etc/group 啰～
-l  ：后面接账号名称。亦即是修改账号名称， /etc/passwd 的第一栏！
-s  ：后面接 Shell 的实际档案，例如 /bin/bash 或 /bin/csh 等等。
-u  ：后面接 UID 数字啦！即 /etc/passwd 第三栏的资料；
-L  ：暂时将使用者的密码冻结，让他无法登入。其实仅改 /etc/shadow 的密码栏。
-U  ：将 /etc/shadow 密码栏的 ! 拿掉，解冻啦！
范例：

范例一：修改使用者 dmtsai 的说明栏，加上『VBird's test』的说明。
[root@linux ~]# usermod -c "VBird's test" dmtsai
[root@linux ~]# grep dmtsai /etc/passwd
dmtsai:x:501:501:VBird's test:/home/dmtsai:/bin/bash

范例二：使用者 dmtsai 密码在 2006/01/01 失效。
[root@linux ~]# usermod -e "2006-01-01" dmtsai
[root@linux ~]# grep dmtsai /etc/shadow
dmtsai:$1$24ISJM4K$bbdijdreoieaVaBMAHsm6.:13026:0:99999:7::13149:

范例三：暂时冻结 dmtsai 的密码！
[root@linux ~]# usermod -L dmtsai
[root@linux ~]# grep dmtsai /etc/shadow
dmtsai:!$1$24ISJM4K$bbdijdreoieaVaBMAHsm6.:13026:0:99999:7::13149:
# 注意到，密码栏(第二栏)多了一个 ! 号！那个惊叹号会让密码无效喔！
[root@linux ~]# usermod -U dmtsai  <==这样就解开了！

范例四：万一 dmtsai 这个家伙被建立时忘记建立家目录，该如何是好？
[root@linux ~]# usermod -d /home/dmtsai2 -m dmtsai
# 如果仅是 -d /home/dmtsai2 表示仅修改 /etc/passwd 第六栏的内容而已，
# 如果加上 -m 这个参数，则表示新建一个家目录的意思！
# 另外，如果原本的家目录是 /home/dmtsai ，那 -d /home/dmtsai2 -m
# 会将原本的 /home/dmtsai 更名为 /home/dmtsai2 喔！
```

usermod 是系统管理员 root 用来管理账号身份的相关数据的，不过，这个 usermod 程序的功能其实也被很多其它的指令所取代喔！例如 chfn 与 chsh 等等的～ 不过，无论如何，您还是可以用 usermod 来微调使用者账号的相关资料啦！

- userdel

这个功能就太简单了～目的在删除使用者啦～与他相关的档案有：

- /etc/passwd
- /etc/shadow
- /home/username

整个指令的语法是：

```
[root@linux ~]# userdel [-r] username
参数：
-r  ：连同使用者的家目录也一起删除
范例：

范例一：删除 vbird2 ，连同家目录一起删除
[root@linux ~]# userdel -r vbird2
```

这个指令下达的时候要小心了！通常我们要移除一个账号的时候，你可以手动的将 /etc/passwd 与 /etc/shadow 里头的该账号取消即可！一般而言，如果该账号只是『 暂时不启用』的话，那么将 /etc/shadow 里头最后倒数一个字段设定为 0 就可以让该账号无法使用，但是所有跟该账号相关的数据都会留下来！使用 userdel 的时机通常是『 你真的确定不要让该用户在主机上面使用任何数据了！』

另外，其实使用者如果在系统上面操作过一阵子了，那么该使用者其实在系统内可能会含有其它档案的。举例来说，他的邮件信箱 (mail box) 或者是例行性命令 (crontab) 之类的档案。 所以，如果想要完整的将某个账号完整的移除，最好可以在下达 userdel -r username 之前， 先以『 find / -user username 』查出整个系统内属于 username 的档案， 然后再加以删除吧！

使用者功能：chfn, chsh

不论是 useradd/usermod/userdel ，都是系统管理员所能够使用的指令， 如果我是一般身份使用者，那么我是否除了密码之外，就无法更改其它的数据呢？ 当然不是啦！这里我们介绍两个一般身份使用者常用的账号数据变更指令啰！

- chsh

```
[dmtsai@linux ~]$ chsh [-ls]
参数：
-l  ：列出目前系统上面可用的 shell ，其实就是 /etc/shells 的内容！
-s  ：设定修改自己的 Shell 啰
范例：

范例一：列出目前系统上面所以的 shell ，并且指定 csh 为自己的 shell
[dmtsai@linux ~]$ chsh -l
/bin/sh
/bin/bash
/sbin/nologin
/bin/ksh
/bin/tcsh
```

```
/bin/csh
/bin/zsh
[dmtsai@linux ~]$ chsh -s /bin/csh; grep dmtsai /etc/passwd
Password:  <==为了防止账号被乱搞～所以需要输入 dmtsai 的密码确认!
Shell changed.
dmtsai:x:501:501::/home/dmtsai:/bin/csh
```

这个指令重点就是在更改使用者的 shell 啰～如上所述，我就可以修订好 dmtsai 的 shell 啦!

- chfn

```
[root@linux ~]# chfn [-foph]
参数：
-f  ：后面接完整的大名；
-o  ：您办公室的房间号码；
-p  ：办公室的电话号码；
-h  ：家里的电话号码!
范例：

范例一：我用 dmtsai 这个使用者来更改一下自己的相关信息!
[dmtsai@linux ~]$ chfn
Changing finger information for dmtsai.
Password: <==为了防止账号被乱搞～所以需要输入 dmtsai 的密码确认!
Name []: VBird’ Test account
Office []: Tainan office 1
Office Phone []: 06-1234567
Home Phone []: 06-7654321

Finger information changed.
[dmtsai@linux ~]$ grep dmtsai /etc/passwd
dmtsai:x:501:501:VBird’ Test account,Tainan office 1,06-1234567,06-7654321:
/home/dmtsai:/bin/bash
```

这个指令说实在的，除非是你的主机有很多的用户，否则倒真是用不着这个程序!这就有点像是 bbs 里头更改你『个人属性』的那一个资料啦!这个程序主要都是搭配 finger 这支程序在运作的!不过，由于 finger 这支程序不是很安全，所以预设是没有安装他的! 如果您想要玩一下 finger 的话，那么请先参考 RPM 套件安装内容后，在安装 finger 的 RPM 档案，然后再来玩吧!底下这里鸟哥还是先简单的介绍一下就好了!

使用 chfn 这个指令之后，程序会要求您输入许多的信息，包含了：

- 密码
- 昵称
- 办公室号码
- 办公室电话
- 家里电话

不过，这些信息其实更改的都是原本的 /etc/passwd 里面的第五栏说明数据啦！ 每个信息中间都以逗号『,』分隔开来而已。如上所示， dmtsai 的说明栏救被更动过啰！ ^_^

- finger

finger 的中文字面意义是：『手指』，嘿嘿！这个 finger 可以查阅的数据可就多了！ 刚刚我们不是使用 chfn 来修改 dmtsai 这个使用者的相关信息吗？那些个相关信息就可以利用 finger 来查阅出来的！他的查询方法如下：

```
[root@linux ~]# finger [-s] username
参数：
-s  ：使用长串数据输出格式。
范例：

范例一：将刚刚 dmtsai 建立的一些使用者信息呼叫出来视察！
[root@linux ~]# finger dmtsai
Login: dmtsai                        Name: VBird's Test account
Directory: /home/dmtsai              Shell: /bin/bash
Office: Tainan office 1, 06-1234567     Home Phone: 06-7654321
Last login Tue Aug 30 15:01 (CST) on tty1 from localhost
No mail.
No Plan.
```

有趣吧！这个 finger 还可以用来查询别部主机的账号呢！不过，目前通常用在本机账号的查询。 因为 finger 算是比较危险的指令，所以，有些 linux distributions 预设是不安装他的， 不过，如果您按照鸟哥说明的方式来完整安装 FC4 的话，那就没有问题的啦！可以操作的。

不过，你或许会觉得有趣的是，怎么 finger 的结果最底下显示『No mail. No Plan.』呢？ 呵呵！ finger 会主动去 /var/spool/mail 查询看看有没有该账号的邮件信箱 (mailbox)，而且还会去查询 ~/.plan 那个档案，那就是计划档啦～比如说，我在 dmtsai 家目录底下建立 .plan 这个档案，他的内容是『DmTsai will write something....』，结果使用 finger 时， 嘿嘿！您可以自行看看结果会怎样啊！ ^_^

- id

id 这个指令则可以查询某人或自己的相关 UID/GID 等等的信息，他的参数也不少，不过， 都不需要记～反正使用 id 就全部都列出啰～ ^_^

```
[root@linux ~]# id [username]

范例一：查阅自己的相关信息！
[root@linux ~]# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),10(wheel)

范例二：查阅一下 dmtsai 吧～
[root@linux ~]# id dmtsai
uid=501(dmtsai) gid=501(dmtsai) groups=501(dmtsai),100(users)
```

再次强调一下，那个 groups 指的是目前该使用者所属的所有群组，但是您必须要了解什么是『 初始群组与有效群组 』的差异喔！

新增与移除群组

OK！了解了账号的新增、删除、更动与查询后，再来我们可以聊一聊群组的相关内容了。 基本上，群组的内容都与这两个档案有关：

- /etc/group
- /etc/gshadow

群组的内容其实很简单，都是上面两个档案的新增、修改与移除而已， 不过，如果再加上有效群组的概念，那么 newgrp 与 gpasswd 则不可不知呢！

- groupadd

```
[root@linux ~]# groupadd [-g gid] [-r]
参数：
-g  ：后面接某个特定的 GID ，用来直接给予某个 GID ～
-r  ：建立系统群组啦！与 /etc/login.defs 内的 GID_MIN 有关。
范例：

范例一：新建一个群组，名称为 group1
[root@linux ~]# groupadd group1
[root@linux ~]# grep group1 /etc/group /etc/gshadow
/etc/group:group1:x:502:
/etc/gshadow:group1:!::
# 注意注意！在 /etc/gshadow 里面可以发现，密码是不许登入的喔！

范例二：新建一个系统群组，名称为 group2
[root@linux ~]# groupadd -r group2
[root@linux ~]# grep group2 /etc/group /etc/gshadow
/etc/group:group2:x:101:
/etc/gshadow:group2:!::
```

了解 -r 有没有的差异了吗？！是的～结果会跟 /etc/login.defs 里面的设定有关喔！ 而且以 groupadd 新增的账号，预设都不能使用密码的方式登入的～ 也就是说，预设是私有群组，并无法使用 newgrp 来登入的呢！

- groupmod

跟 usermod 类似的，这个指令仅是在进行 group 相关参数的修改而已。

```
[root@linux ~]# groupmod [-g gid] [-n group_name]
参数：
```

```
-g  ：修改既有的 GID 数字；
-n  ：修改既有的群组名称
范例：

范例一：将刚刚上个指令建立的 group2 名称改为 groupname ， GID 为 103
[root@linux ~]# groupmod -g 103 -n groupname group2
[root@linux ~]# grep groupname /etc/group /etc/gshadow
/etc/group:groupname:x:103:
/etc/gshadow:groupname:!::
```

不过，还是那句老话，不要随意的更动 GID ，容易造成系统资源的错乱喔！

- groupdel

呼呼！ groupdel 自然就是在删除群组的啰～用法很简单：

```
[root@linux ~]# groupdel [groupname]

范例一：将刚刚的 groupname 删除！
[root@linux ~]# groupdel groupname

范例二：若要删除 dmtsai 这个群组的话？？
[root@linux ~]# groupdel dmtsai
groupdel: cannot remove user's primary group.
```

为什么 groupname 可以删除,但是 dmtsai 就不能删除呢？原因很简单，『有某个账号 (/etc/passwd) 的 initial group 使用该群组！』 如果查阅一下，你会发现在 /etc/passwd 内的 dmtsai 第四栏的 GID 就是 /etc/group 内的 dmtsai 那个群组的 GID ，所以啰，当然无法删除～否则 dmtsai 这个使用者登入系统后， 就会找不到 GID ，那可是会造成很大的困扰的！那么如果要删除 dmtsai 这个群组呢？ 你『必须要确认 /etc/passwd 内的账号没有任何人使用该群组作为 initial group 』才行喔！所以，你可以：

- 修改 dmtsai 的 GID ，或者是：
- 删除 dmtsai 这个使用者。

- gpasswd

除了设定群组之外，我们还可以针对系统上面有的群组进行一些『密码』的给予喔！ 这个密码给予之后，该群组就能够让某些人登入成为有效群组呢！挺有趣的。 另外，如果系统管理员太忙了，无法针对每个群组来管理，那么『 系统管理员还可以将某位使用者设定成为该群组的团长喔！ 』 很有趣吧～虽然目前比较少人这么玩了，不过，鸟哥在这里还是跟大家介绍介绍吧！

```
关于系统管理员(root)做的动作：
[root@linux ~]# gpasswd groupname
[root@linux ~]# gpasswd [-A user1,...] [-M user3,user4...] groupname
[root@linux ~]# gpasswd [-rR] groupname
参数：
```

```
    ：若没有任何参数时，表示给予 groupname 一个密码(/etc/gshadow)
-A  ：将 groupname 的主控权交由后面的使用者管理(该群组的管理员)
-M  ：将某些账号加入这个群组当中！
-r  ：将 groupname 的密码移除
-R  ：让 groupname 的密码栏失效，所以 newgrp 就不能使用了！

关于群组管理员(Group administrator)做的动作：
[someone@linux ~]$ gpasswd [-ad] user groupname
参数：
-a  ：将某位使用者加入到 groupname 这个群组当中！
-d  ：将某位使用者移除出 groupname 这个群组当中。

范例一：建立一个新群组，名称为 testgroup 且群组交由 dmtsai 管理：
[root@linux ~]# groupadd testgroup
[root@linux ~]# gpasswd testgroup
Changing the password for group testgroup
New Password:
Re-enter new password:
# 输入两次密码就对了！
[root@linux ~]# gpasswd -A dmtsai -M dmtsai,vbird testgroup
[root@linux ~]# grep testgroup /etc/group /etc/gshadow
/etc/group:testgroup:x:502:dmtsai,vbird
/etc/gshadow:testgroup:1CEVbrcjxO6Ps:dmtsai:dmtsai,vbird
# 很有趣吧！此时 dmtsai 则拥有 testgroup 的主控权喔！若以我们讨论区 (
# http://phorum.vbird.org 的概念来说，群组管理员有点像『版主』啦！

范例二：以 dmtsai 登入系统，并且让他加入 vbird1 成为 testgroup 成员之一：
[dmtsai@linux ~]$ gpasswd -a vbird1 testgroup
Adding user vbird1 to group testgroup
```

很有趣的一个小实验吧！我们可以让 testgroup 成为一个可以公开的群组， 然后建立起群组管理员，群组管理员可以有多个。在这个案例中， 我将他设定为 dmtsai ，所以， dmtsai 就可以自行增加群组成员啰～ 呼呼！然后，该群组成员就能够使用 newgrp 啰～

- newgrp

还有印象吗？我们前面谈到 /etc/gshadow 时就提过这个指令了！『newgrp 会额外以另一个 login 来提供使用者登入到另一个 shell 中， 并且将有效群组改为 newgrp 后面接的那个群组，若没有接群组，则预设群组为 initial group 』

密码管理： passwd

再来跟大家提一提那个重要的密码概念！您得要特别留意的是，今天，您的主机若是遭到入侵， 对方的第一个入侵点自然就是您主机上面账号的『密码』了，所以， 如果您的密码定义的比较严格的话，那么自然对方就不容易猜到你的密码，自然就会比较有保障啦！

目前一些 Cracker 较常使用的密码破解软件，大抵是『字典攻击法』 及所谓的『暴力破解法』，就字面上的意义来说，『字典攻击法』是将字典里面所查的到的任何单字或词组都输入的程序中， 然后使用该程序一个一个的去尝试破解你的密码，不要觉得这样的速度似乎很慢，实际上， 现今的计算机运算速度太高了，字典攻击法的操作效率基本上是很高的！另一个『 暴力破解法』就是直接使用键盘上面任何可以使用的按键，然后依照组合，以 1 个, 2 个, 3 个…. 密码组合的方式去破解你的密码！这个方式就真的比较慢一点，如果你的密码组合是 6~8 个字符以上，那么暴力攻击法还是需要好长一段时间才能够破解的了的！

由上面的『字典攻击法』与『暴力破解法』猜测你的密码的方式来说，您知道如何设定一个好的密码了吗？是的，您的密码最好需要底下几个特性：

- 密码中含有数个特殊字符，例如 $#@^&* 及数字键等等：如同上面提到的， 您的按键越奇怪，那么对方就越不容易使用既有的软件来破解！
- 英文字母大小写混合使用；
- 密码长度至少要到 6 ~ 8 个以上才好；
- 没有特殊意义的字母或数字组合，并且夹着很多的特殊字符！

这种密码真的很不容易被破解，但是很不幸的，也很容易被你我忘记！^_^。所以呢， 建议您常常使用一些对别人来说是没有意义，但是对您确有特殊涵意的字眼！ 例如鸟哥常常提到的，我爱我老婆！『 I&Mywife*^』之类的密码！不容易被猜，也挺容易被你自己记住的！ 那么有没有『很要命的密码』呢？有的，底下几种密码就很要命：

- 常用的英文单字：例如 party, park, andyliu, linux, paper 等等，都不好！容易被字典攻击法破解！
- 身边人物的名字，例如配偶、小孩的名字等等， Tom, andy, eric 等等，都不好！
- 单纯的日期：例如您的生日啦！等等的，都不够好！
- 任何与您相关的数字或其它信息，例如身份证号、银行账号等；

VBird 曾经见过直接以账号做为密码的状况！真是要命～太好猜了！

好了！知道了密码的重要性，与基本的设定之后，接着下来我们谈一谈如何手动设定密码吧！基本上， root 可以设定『任何样式的密码』，而且， root 也可以帮助 user 订定他们的密码！至于 user 仅能修改自己的密码！那么修改密码使用什么命令？就是 passwd 这个命令啦！咦！这里突然给他想到几个重要信息，大家赶紧复习一下：

- 如何寻找 passwd 这个指令？
  使用 which passwd 即可
- 如何察看 passwd 这个档案的属性？并请说明他的属性为何？
  使用 ls -l `which passwd` 即可！他具有 SUID 的属性！
- 什么是 SUID ？
  就是该程序在被执行的过程中，具有程序拥有者的权限！
- 我该如何查询 /etc/passwd 与 /usr/bin/passwd 的用法与架构？
  分别使用 man passwd 及 man 5 passwd

这些指令与意义如果都还没有忘记！恭喜您了！真是不错！好了，还记得我们密码放在哪里吗？对啦！就是 /etc/shadow 里面，那个档案的权限是 -rw------- 所以只有 root 可以修改，因此， passwd 必需要具有 SUID 才能让一般使用者修改他们的密码啰！关于 passwd 的用法， 我们前面已经稍微提过一些啰，在底下我们则针对 root 谈一下 passwd 还有什么好功能？？

- passwd

```
[root@linux ~]# passwd [-lunxwS] username
参数：
-l  ：将 username 这个账号的密码锁住 (lock)，在 /etc/shadow 内的密码栏修订～
-u  ：将 -l 的 lock 解开！
-n  ：后面接天数 (数字) ，最短天数；亦即是 /etc/shadow 内的第四栏；
-x  ：后面接天数 (数字) ，最长天数；亦即是 /etc/shadow 内的第五栏；
-w  ：后面接天数 (数字) ，警告天数；亦即是 /etc/shadow 内的第六栏；
-S  ：显示目前这个 username 的相关信息。
范例：

范例一：将 dmtsai 这个使用者的密码冻结，并观察他！
[root@linux ~]# passwd -l dmtsai
Locking password for user dmtsai.
passwd: Success
[root@linux ~]# passwd -S dmtsai
Password locked.
[root@linux ~]# grep dmtsai /etc/shadow
dmtsai:!!$1$TDy6D7eg$jVJV/FMaQn14v5K17sqw6/:13026:0:99999:7::13149:

范例二：将上述密码冻结解开
[root@linux ~]# passwd -u dmtsai
```

其实这个 passwd 指令还挺多用的～尤其很多功能仅有 root 才能执行。 您可以使用 passwd -l 及 passwd -u 来强制让一个使用者『暂时』无法使用该账号， 很方便的啦！ ^_^

使用者身份切换：

什么？在 Linux 系统当中还要作身份的变换？这是为啥？

- 系统平日操作的好习惯：
  事实上，为了安全的缘故， 我们大家都会建议您，操作 Linux 时，尽量以一般身份使用者来操作，等到需要设定系统环境时， 才变换身份成为 root 来进行系统管理，相对比较安全啦！避免作错一些严重的指令～～

- 用较低权限启动系统服务
  相对于系统安全，有的时候，我们必须要以某些系统账号来进行程序的进行。 举例来说， Linux 主机上面的一套软件，名称为 apache ，我们可以额外建立一个名为 apache 的使用者来启动 apache 啊，如此一来，如果这个程序被攻破，至少系统还不至于就损毁了～

- 软件本身的限制
  这里有个很有趣的问题要来跟大家分享一下，还记得在古老的年代里面，还没有 ssh 的时候，我们都是使用 telnet 登入系统的，偏偏系统预设是不开启 root 以 telnet 登入，那么好了！我们要怎样远程操控我们的 Linux 主机呀！？ 因为由前面的介绍我们不难发现，系统当中最特殊的账号就是 UID 为 0 的使用者了，他具有至高无上的权力！而且是系统管理员必须要具备的身份， 否则怎样操控主机呢？您说是吧！好了，那么 telnet 将 root 的登入权限关掉了，而如果我们在制作一个使用者，并将其 UID 变为 0 的话又如何？嘿嘿！很抱歉， telnet 就是认 UID 的，所以肯定还是进不了系统， 这个时候要怎么办呀！？就是变换身份呀！将一般使用者的身份变成了 root 就行了！

但是怎样变换身份呀？怎么说呢？就是说，一般而言，我们都不希望以 root 的身份登入主机，以避免被怪客入侵了！但是一部主机又不可能完全不进行修补或者是设定等动作！ 这个时候要如何将一般使用者的身份变成 root 呢？主要有两种方式，分别是：

- 以 su 直接将身份变成 root 即可，但是这个指令却需要 root 的密码，也就是说，如果你要以 su 变成 root 的话，你的一般使用者就必须要有 root 的密码才行；

- 所以当有很多人同时管理一部主机的时候，那么 root 的密码不就很多人知道了？不是很好吧？所以，如果不想要将 root 的密码流出去呢？呵呵！可以使用 sudo 来进行工作呦！

底下我们就来说一说 su 跟 sudo 的用法啦！

---

su

```
[root@linux ~]# su [-lcm] [username]
参数：
-   ：如果执行 su - 时，表示该使用者想要变换身份成为 root ，且使用 root 的
      环境设定参数档，如 /root/.bash_profile 等等。
-l  ：后面可以接使用者，例如 su -l dmtsai ，这个 -l 好处是，可使用欲变换身份者
      他的所有相关环境设定档。
-m  ：-m 与 -p 是一样的，表示『使用目前的环境设定，而不重新读取新使用者的设定档。』
-c  ：仅进行一次指令，所以 -c 后面可以加上指令喔！
范例：

范例一：由原本的 dmtsai 这个使用者，变换身份成为 root 。
[dmtsai@linux ~]$ su
Password: <==这里输入 root 的密码喔！
[root@linux ~]# env
USER=dmtsai
USERNAME=root
MAIL=/var/spool/mail/dmtsai
LOGNAME=dmtsai
# 注意到了吗？如果使用 su 没有加上 - 的话，那么很多原本使用者的相关设定会继续存在，
```

```
# 这也会造成后来的 root 身份在执行时的困扰。最常见的就是 PATH 这个变量的问题！

[root@linux ~]# exit  <==这样可以离开 su 的环境！
[dmtsai@linux ~]$ su -
Password: <==这里输入 root 的密码喔！
[root@linux ~]# env
USER=root
MAIL=/var/spool/mail/root
LOGNAME=root
# 了解差异了吧？！所以，下次在变换成为 root 时，记得最好使用 su - 喔！

范例二：使用 root 的身份，执行 head -n 3 /etc/shadow
[dmtsai@linux ~]$ su - -c "head -n 3 /etc/shadow"
Password: <==这里输入 root 的密码喔！
root:$1$jaldj9843u29jlj9u839jljlcghjlE/:12959:0:99999:7:::
bin:*:12959:0:99999:7:::
daemon:*:12959:0:99999:7:::

范例三：原本是 dmtsai 这个使用者，想要变换身份成为 vbird 时？
[dmtsai@linux ~]$ su -l vbird
Password: <==这里输入 vbird 的密码喔！
```

这个 su 指令可以让你在不同的使用者之间切换身份，当 su 后面没有加上使用者账号时， 那么预设就是以 root 作为你切换的那个身份啦！其实，这个指令最大的用途也是在这里！ 就是让一般使用者变成 root 啦！而要特别留意的则是 su 的使用方式上， 由于『是否读入欲切换的身份者的环境参数档案』的不同，所以您必须要留意喔！

- 如果只是想要使用 root 的身份来操作系统，但是原有的环境参数并不想要改变， 那么可以使用『 su 』直接切换身份成为 root ，例如上面的范例一所示。 此时， MAIL/PATH/USER 等环境变量都还是原来那位登入者喔！所以要特别留意例如 PATH 这个可能影响到执行指令进行的变量才行！

- 如果您想要保留原有的环境参数，那么环境变量当中，最麻烦的当属 PATH 这个东西， 由于为了避免一般使用者使用了 root 的管理指令，所以通常 Linux 都会将指令分类放在两个主要的目录，分别是 /bin 与 /sbin ！那个 /sbin 大多是 super user 就是 root 用来管理系统的指令啦！所以，可能的话，将你习惯操作的那个账号的 PATH 重新设定成为 root 的 PATH ，这样也比较方便呀！

- 无论如何，还是建议您如果要切换成为某个身份，使用『 su - 』或者是『 su -l usename 』 会比较好一点～否则容易造成环境变量的差异～

- 另外，如果仅想要执行一次 root 的指令，那么可以参考 -c "command" 这种 su 的使用方式喔！

- 当 root 使用 su 切换身份时，他并不需要输入密码喔！

虽然使用 su 很方便啦～不过，缺点是当我有很多管理员时，那么是否每个人都需要知道 root 的密码？这样很危险ㄟ！ root 的密码可能会外流～怎么办？没关系，我们可以使用 sudo 来取代 su 喔。

---

sudo

使用 su 切换身份真的是很简单啦～不过， su 却有一个很严重的问题， 那就是.....我们必须要知道想要变成的那个人的登入密码～ 举例来说，如果我想要变成 root ，那么就必须要知道 root 的密码才行，如果我想要变成 dmtsai 来工作，那么除非我是 root ，否则就必须要知道 dmtsai 这个使用者的密码才行～而众所皆知的，如果多人管理一部主机的话，大家都知道 root 的密码，那......挺危险的，不是吗？！

这个时候， sudo 就派的上用场啰～那么 sudo 是怎样工作的呢？

- 当使用者执行 sudo 时，系统会主动的去寻找 /etc/sudoers 档案，判断该使用者是否有执行 sudo 的权限；
- 若使用者具有可执行 sudo 的权限后，便让使用者『输入使用者自己的密码』来确认；
- 若密码输入成功，便开始进行 sudo 后续接的指令；
- 不过， root 执行 sudo 时，不需要输入密码；
- 若欲切换的身份与执行者身份相同，那也不需要输入密码。

要注意的是，使用者『输入的是自己的密码，而不是欲切换成为他的那个身份的密码！』 举例来说，假设 dmtsai 具有执行 sudo 的权限，那么当他以 sudo 执行 root 的工作时， 他需要输入的是 dmtsai 自己的密码，而不是 root 的密码！嘿嘿！很棒吧！ ^_^ 如此一来，大家可以使用自己的密码执行 root 的工作，而不必知道 root 的密码，安全多了。 此外，使用者能够执行的指令是可以被限制的！ 所以，我们可以设定 dmtsai 仅能进行 shutdown 的工作，或者是其它一些简单的指令， 嘿嘿！是否很棒啊！

不过，由上面的说明当中，您也会了解，是否具有 sudo 的执行权限是很重要的， 而 sudo 的执行权限与 /etc/sudoers 这的档案有关。在预设的情况下，只有 root 才能够使用 sudo 呢！至于编辑 /etc/sudoers 则需要 visudo 这个指令。好了， 底下我们就来看一看 sudo 的语法先。

```
[root@linux ~]# sudo [-u [username|#uid]] command
参数：
-u  ：后面可以接使用者账号名称，或者是 UID。例如 UID 是 500 的身份，可以：
      -u #500 来作为切换到 UID 为 500 的那位使用者。
范例：

范例一：一般身份使用者使用 sudo 在 /root 底下建立目录：
[dmtsai@linux ~]$ sudo mkdir /root/testing
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.
```

```
Password: <==这里输入 dmtsai 自己的密码
dmtsai is not in the sudoers file.  This incident will be reported.
# 瞧！因为 dmtsai 不在 /etc/sudoers ，所以他就无法执行 sudo 喔！

范例二：假设 dmtsai 已经具有 sudo 的执行权限，如何在 /root 底下建立目录？
[dmtsai@linux ~]$ sudo mkdir /root/testing
Password: <==这里输入 dmtsai 自己的密码

范例三：如何将 sudo 与 su 搭配使用？
[dmtsai@linux ~]$ sudo su -

范例四：dmtsai 想要切换身份成为 vbird 来进行 touch 时？
[dmtsai@linux ~]$ sudo -u vbird touch /home/vbird/test
```

上面我进行了四个范例，不过，要注意的是，若我是以 dmtsai 来进行的， 那么除了第一次执行 sudo 需要输入密码，未来的时间内(在这次登入的状况中)就不需要再重复输入密码了！ 呼呼！真是很人性啊的设计啊～ ^_^

上面这四个范例我都是以 dmtsai 这个使用者来进行的，但是，在预设的情况中， 您的使用者应该是不能使用 sudo 的～这是因为我们上面提到的啊，还没有去设定 /etc/sudoers 嘛！ 所以啰，如果您要测试上面的范例之前，是需要将 /etc/sudoers 动动手脚的。 不过，因为 /etc/sudoers 需要一些比较特别的语法，因此，如果你直接以 vi 去编辑他时， 如果输入的字句错误，可能会造成无法启用 sudo 的困扰，因此，建议您一定要使用 visudo 去编辑 /etc/sudoers 喔！(注：visudo 必须要使用 root 的身份来执行！)

```
[root@linux ~]# visudo
# sudoers file.
# This file MUST be edited with the 'visudo' command as root.
# See the sudoers man page for the details on how to write a sudoers file.
#
# Host alias specification
# User alias specification
# Cmnd alias specification
# Defaults specification
# Runas alias specification
# User privilege specification
root    ALL=(ALL) ALL
dmtsai  ALL=(ALL) ALL   <==这里将 dmtsai 制作成完全可用！

# Uncomment to allow people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL
# Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL
# Samples
# %users  ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users  localhost=/sbin/shutdown -h now
```

使用 visudo 之后，其实就会出现一个 vi 的画面啦！他就是以 vi 来开启 /etc/sudoers ， 不过，当我们储存离开时， visudo 会额外去检查 /etc/sudoers 内部的语法， 以避免使用者输入错误的信息了。我上面只有加入一行，就是让那个 dmtsai 成为可以随意使用 sudo 的身份而已。基本上， /etc/sudoers 的结构您可以使用 man sudoers 去查阅， 该说明内容说的很清楚，而且还有一些范例呢！鸟哥在这里仅作一些简单的说明就是了。 那一行『 dmtsai ALL=(ALL) ALL 』代表的意义是：

使用者账号 登入的主机 = (可以变换的身份) 可以下达的指令

因此，我上面这一行的意义是：『 dmtsai 这个使用者，不论来自何方， 他可以变换成任何 Linux 本机上面有的所有账号，并执行所有的指令』的意思啦！ 假如您的系统里面，有个 Web 的软件是以 www 这个使用者来进行编辑的， 您想要让 vbird2 这个使用者可以用 www 这个账号进行编辑，那么就应该写成：

vbird2 ALL = (www) ALL

如果错写成：

vbird2 ALL = ALL

亦即没有加上身份宣告的话，那么『预设是仅能进行 root 的身份切换』而已喔！ ^_^ 这可是很重要的一个观念呢！另外，如果想要以使用者的群组来进行规范的话，那么在 『使用者账号』的字段，前面加上『 % 』时，就代表是群组 (group) 的身份了。 举例来说，我想要让系统里面所有属于 wheel 这个群组的使用者都能够进行 sudo 时，可以这样写：

%wheel ALL = (ALL) ALL

而如果你还想要让这个群组内的使用者在使用 sudo 时，不需要输入密码， 那么可以在『可以下达的指令』那个字段内多加入一个参数，名为『NOPASSWD:』即可，亦即：

%wheel ALL = (ALL) NOPASSWD: ALL

另外，除了单一个人或单一群组之外，我们还可以额外指定一些『账号别名、主机别名、指令别名』 等等的数据来相互套用喔！真是好棒啊！不过，关于别名的使用上，『必须要使用大写字符』才行喔！ 好了，我们来做一些练习，让您可以很清楚的知道如何进行 visudo 的设定吧！

例题:我想要建立一个可以帮忙系统管理员变更使用者密码的群组,名称为 ADMPW (注意,在 sudoers 内，这个别名的名称一定要是大写字符才行！)但是这个群组不能修改 root 的密码喔！且他们执行 sudo 时，不需要密码验证。

答：

我以 root 的身份使用 visudo ，进入编辑画面后，去设定成底下的模样：

```
User_Alias ADMPW = vbird, dmtsai, vbird1, vbird3
ADMPW ALL = NOPASSWD: !/usr/bin/passwd, /usr/bin/passwd [A-Za-z]*, \
                      !/usr/bin/passwd root
```

上面的意思是说，我的系统上面有四个账号，分别是 vbird, vbird1, vbird3 与 dmtsai 这四个账号加入 sudo 内的 ADMPW 群组中，这四个账号可以使用 sudo 进行『 /usr/bin/passwd＊ 』密码的更改动作，但是不能 (在指令前面加入 ! 代表不可) 使用 /usr/bin/passwd 或 /usr/bin/passwd root ，如此一来，就让该 ADMPW 可以更改使用者的密码，但是不能变更 root 的密码啰！ ^_^

在 /etc/sudoers 里头加入别名有很多的好处，举例来说，以上面的例子来讲， 假设未来我有其它的使用者要加入该密码管理的群组时，直接将账号加入 ADMPW 那个群组中就好了，很简单的使用吧！ ^_^。再看看下一题：

例题：我的系统中有 DNS 服务，他的启动指令在 /etc/init.d/named ，如果我想要建立一个

DNSMASTER 的群组来管理他时？如何是好？
答：

我以 root 的身份使用 visudo ，进入编辑画面后，去设定成底下的模样：

```
User_Alias DNSMASTER = vbird, dmtsai
Cmnd_Alias DNSCMD = /etc/init.d/named, /usr/bin/vim /var/named/*
DNSMASTER ALL = DNSCMD
```

看的懂吗？嘿嘿！因为 DNS 的设定档大多在 /var/named 里面，所以，我也允许相关账号用 vi 去处理 DNS 的设定档啦！很简单对吧！ ^_^

好了，我们知道 sudo 可以搭配 su 来进行一堆系统的工作对吧！ 因为 sudo 仅能进行一次指令，很麻烦，如果我能够将 sudo 与 su 搭配在一起，不就很棒了吗？ 这个时候，我可以利用上面已经建立好的 ADMPW 群组来新增这一行：

```
ADMPW ALL = /bin/su
```

如此一来，在 ADMPW 内的使用者，就可以利用『 sudo su - 』来切换身份成为 root 啰～ 真是棒得不得了啊！ ^_^

使用者的特殊 shell 与 PAM 模块

我们前面一直谈到的大多是一般身份使用者与系统管理员 (root) 的相关操作， 而且大多是讨论关于可登入系统的账号来说。那么换个角度想，如果我今天想要建立的， 是一个『仅能使用 mail server 相关邮件服务的账号，而该账号并不能登入 Linux 主机』呢？ 如果不能给予该账号一个密码，那么该账号就无法使用系统的各项资源，当然也包括 mail 的资源， 而如果给予一个密码，那么该账号就可能可以登入 Linux 主机啊！呵呵～伤脑筋吧～ 所以，底下让我们来谈一谈这些有趣的话题啰！

特殊的 shell, /sbin/nologin

如果你曾经仔细的看过 /etc/shells 这个系统可用的 shell 档案，以及 /etc/passwd 这个档案的内容时，你应该会发现，嘿嘿！怎么有个怪怪的 /sbin/nologin 啊！ 这是什么 shell 呢？呵呵！赶紧利用 man nologin 就可以知道啦～

其实，这个 shell 通常是给系统账号使用的，因为这个 /sbin/nologin 事实上并无法给予账号实际登入，如果你利用 usermod 修改了 dmtsai 这个使用者的 shell 成为 /sbin/nologin 之后， 再次想要以 dmtsai 重新登入系统时，他在屏幕上会出现这样的讯息：

```
This account is currently not available.
```

嘿嘿！它说的是『这个账号并不能被允许登入啦！』不过，这个账号却可以进行其它的工作喔！ 举例来说，各个系统账号，打印工作由 lp 这个账号在管理， WWW 服务由 apache 这个账号在管理， 他们都可以进行系统程序的工作，但是『就是无法登入主机』而已啦！ ^_^

换个角度来想，如果我的 Linux 主机提供的是邮件服务，所以说，在这部 Linux 主机上面的账号， 其实大部分都是用来收受主机的信件而已，并不需要登入主机的呢！ 这个时候，我们就可以考虑让单纯使用 mail 的账号以 /sbin/nologin 做为他们的 shell ， 这样，最起码当我的主机被尝试想要登入系统时，可以拒绝该账号呢！

另外，如果我想要让某个具有 /sbin/nologin 的使用者知道，他们不能登入主机时， 其实我可以建立『 /etc/nologin.txt 』这个档案， 并且在这个档案内说明不能登入的原因，那么下次当这个使用者想要登入系统时， 屏幕上出现的就会是 /etc/nologin.txt 这个档案的内容，而不是预设的内容了！

PAM 模块：/etc/nologin, /etc/securetty

当一个使用者想要登入 Linux 主机时，他受到什么限制呢？我们说， 他除了必须要通过 /etc/passwd 及 /etc/shadow 的验证并取得相关的权限数据， 最后获得一个 shell 之外，事实上，他在登入系统之前，就得要通过 PAM (Pluggable Authentication Modules, 嵌入式模块) 的验证才行。

PAM 模块的用途非常的多，除了可以在使用者登入时进行身份的验证之外， 也可以辅助一些应用程序的验证之用喔！举例来说，我们前面提到的密码修改程序『 passwd 』 ，当我们执行密码修订的时候，这个程序不是会告诉我们您输入的密码是否合于规范吗？ 如果是记录在字典当中的密码，或者是与账号相同的密码，那么就会被 PAM 模块打回票， 也就无法通过验证了！

那么 PAM 怎么运作呢？我们同样以 /usr/bin/passwd 这支程序来作为简单的说明好了：

1. 使用者开始执行 /usr/bin/passwd 这支程序，并输入密码；
2. passwd 开始呼叫 PAM 模块，PAM 模块会搜寻 passwd 程序的 PAM 相关设定档案， 这个设定档一般是在 /etc/pam.d/ 里面的与程序同名的档案，所以，在本例中， PAM 会去搜寻 /etc/pam.d/passwd 这个设定档；
3. 经由 /etc/pam.d/passwd 设定文件的数据，取用 PAM 所提供的相关模块来进行验证；
4. 将验证结果回传给 passwd 这支程序，而 passwd 这支程序会根据 PAM 回传的结果决定下一个动作 (重新输入新密码或者通过验证！)

这个过程提供我们几个重要的信息：

- PAM 的设定档放置在 /etc/pam.d/ 这个目录中；
- 至于更多的环境相关设定则放置在 /etc/security/* 内；
- PAM 是透过自己提供的相关模块来进行验证，模块放置在 /lib/security/* 内。

至于 PAM 相关模块的运作，有兴趣的话，您可以前往您 Linux 主机的： /usr/share/doc/pam* 目录去瞧一瞧，里面有相当多丰富的信息可以提供给你参考。 我们这里仅就使用者登入相关的模块来进行一些简单的说明而已喔。

- PAM 的设定文件设定范例：

反正 PAM 模块就是让程序呼叫用的，而当程序呼叫时， PAM 就会利用相对应的设定档来进行一些验证就是了。我们还是举 passwd 为例好了， 如果你去观察一下 /etc/pam.d/passwd 的内容时，他是这样的：

```
[root@linux ~]# cat /etc/pam.d/passwd
#%PAM-1.0
auth       required     pam_stack.so service=system-auth
```

```
account     required     pam_stack.so service=system-auth
password    required     pam_stack.so service=system-auth
```

基本上，在这个档案内，每一行都是一个动作，而每个动作都分为四个字段，分别是：

验证的类别　验证的控制标准　使用的 PAM 模块　该模块的能使用的参数

验证的类别 (Module type) 共分为四种类，分别说明：

- auth
  这种类别主要用来检验使用者的身份验证，所以这种类别通常是需要密码来检验的。
- account
  这种类别则主要在检验使用者是否具有正确的使用权限，举例来说，当你使用一个过期的密码来登入时， 当然就无法正确的登入了。
- session
  这种类别主要在管理当使用者正确的使用该程序时的环境设定。举例来说，我们登入 Linux 其实使用的是 /bin/login 这个程序的相关功能的，所以，当实际登入后，在操作 shell 的过程中，都是受 session 这种类别的设定所控制的喔！另外，如果使用 session 这种类别时， 则该程序在正式使用之前与使用结束之后，都会有相关纪录被记到登录文件当中喔！
- password
  至于这种类别，则主要在提供验证的修订工作，举例来说，就是修改/变更密码啦！

那么『验证的控制标准(control flag)』又是什么？简单的说，他就是『验证通过的标准』啦！ 总共也有四种方式，分别是：

- required
  当模块设定为这种控制标准时，该模块的验证必须要成功，否则就会回传一个 failure 的讯息。不过，不论此一动作的模块是否成功，接下去的模块都还会继续动作！ 而若有 failure 的讯息时，也会在后续的动作都进行完毕之后，才会回传给原程序。 比底下的 requisite 还要优秀的地方，在于该模块底下的动作可能具有登录文件纪录 (log) 的举动，则错误的讯息才会被纪录起来喔！

- requisite
  当模块设定为 requisite 时，该模块的认证要求同样的需要成功才行。 不过，如果该模块没有通过验证，那么 PAM 会『立刻』回报程序一个 failure 的值， 也就是说，若该次动作的模块后续还有其它模块时，其它模块的动作将不会被启用。

- optional
  这个模块控件目大多是在显示讯息而已，并不是用在验证方面的。

- sufficient
  这个模块控制标准也挺有趣的，相对于 requisite 是『发生错误时，立刻回报原执行程序 failure ， 并且中断 PAM 的运作』， sufficient 则是『顺利通过验证时，立刻回报原程序通过的讯息， 并且中断 PAM 的运作』。呵呵！完全相反喔！

至于 PAM 的模块方面，目前我们的 FC4 提供的 PAM 模块真的够多了， 这些模块实际上都放置在 /lib/security/ 目录中，FC4 相关的 PAM 说明文件则放置在 /usr/share/doc/pam-*/* 里面，您可以根

据每个不同的模块去讨论他的用途， 鸟哥在这里仅针对我们登入时所使用的 login 这个程序的 PAM 设定文件，也就是 /etc/pam.d/login 这个档案的内容来稍做说明：

```
[root@linux ~]# cat /etc/pam.d/login
#%PAM-1.0
auth       required  pam_securetty.so
auth       required  pam_stack.so service=system-auth
auth       required  pam_nologin.so
account    required  pam_stack.so service=system-auth
password   required  pam_stack.so service=system-auth
# pam_selinux.so close should be the first session rule
session    required  pam_selinux.so close
session    required  pam_stack.so service=system-auth
session    optional  pam_console.so
# pam_selinux.so open should be the last session rule
session    required  pam_selinux.so multiple open
```

在我们登入 Linux 的时候，使用到的 login 这个程序时，他使用的 PAM 设定档大多是 required 的控制标准，所以必须要通过上述的几个 PAM 模块的验证后，才能够判定是否登入 Linux 。 需要留意的是，我们会看到 session 的模块类型，这表示我们在实际使用 Linux 的资源之前，以及注销 Linux 主机后，相关的数据都会被纪录到登录文件当中。 嘿嘿！所以啰，如果您仔细的看过 /var/log/messages 与 /var/log/secure ， 就能够发现你的一举一动其实是有被纪录下来的喔！ ^_^

另外，在上面表格当中的模块中，比较有趣的是被鸟哥注明特殊字体的部分， 这两个模块：pam_securetty.so 及 pam_nologin 是挺有趣的喔！ 我们就额外来谈一谈这两个关于登入环境的验证模块吧！

- /etc/securetty

这个 pam_securetty 模块是干嘛用的？其实他最主要的功能就是在预防不安全的登入环境啦！ 而且主要是针对 root 这个使用者的身份喔！ 这个模块在被启用时，会去读取 /etc/securetty 这个档案， 我们『可以将被认定是安全的终端机 (terminal) 环境写入这个档案中， 则 root 仅可以在那几个终端机环境下登入』的啦！

一般来说，我们会认定在主机前面的环境是安全的，而使用网络登入的环境则比较危险。 因此，一般 /etc/securetty 的内容大多是这样：

```
tty1
tty2
tty3
tty4
...
```

而没有 pts/0 这类的网络登入的终端接口。这也就是说， root 仅能经由 tty1 这种终端机登入的啦！支持 login 程序的软件有 telnet 服务与本机前面的 tty1~tty6 的 login， 这也是我们提到的，为何使用传统的 telnet 联机主机联机到 Linux 时，预设无法使用 root 身份登入的主要原因啰。

那么如何克服呢？其实也很简单啦，就将这个模块的验证移除即可！ 主要有两种方式：

- 将 /etc/pam.d/login 内，关于 pam_securetty.so 模块的那一行批注掉；
- 将 /etc/securetty 这个档案移除。

如此一来，当我们使用 telnet 联机到 Linux 主机时，就能够直接使用 root 的身份登入了。 不过，鸟哥不建议这么做喔！不过或许您又会问啦，那为什么我使用 ssh 联机时， 就可以直接使用 root 登入呢？呵呵！这是因为 ssh 没有用到这个模块ㄇㄟ！ 不相信吗？仔细自己去查阅一下 /etc/pam.d/sshd 就知道啦！ ^_^。

- /etc/nologin

那么 pam_nologin 又是在搞什么咚咚啊？其实，这个模块也是在控制使用者登入用的。 不过，这个模块只针对一般身份使用者有效，对 root 是没有效果的。这个模块必须要与 /etc/nologin 搭配使用，注意喔，是 /etc/nologin ，而不是 /etc/nologin.txt ， 这两个档案的用途是不相同的喔！ ^_^。

当 /etc/nologin 档案存在时，则任何一个一般身份账号在尝试登入时， 都仅会获得 /etc/nologin 内容的信息，而无法登入主机。举例来说，当我建立 /etc/nologin ，并且内容设定为『This Linux server is maintaining....』， 那么任何人尝试登入时，都仅会看到上面提到的这段讯息，而且无法登入喔！ 一直要到 /etc/nologin 被移除后，一般身份使用者才能够再次的登入啊！

- /etc/security/*

事实上，更多的 PAM 模块设定信息您可以参考 /etc/security/* 里面的档案设定， 尤其是针对使用者利用 Linux 系统资源的 limits.conf 以及时间的 time.conf 。 我们知道使用者利用系统资源的指令是 ulimit ，那么假如我想要让 dmtsai 仅能存取 10MBytes (10240KBytes) 的档案大小，那么我可以这样做：

```
[root@linux ~]# vi /etc/pam.d/limits.conf
# 新增这两行
dmtsai          hard    fsize   10240
@users          hard    fsize   10240
# 注意到，账号前面加上 @ 表示为『群组！』
```

那么下次 dmtsai 或者是属于 users 的群组的使用者登入这个 Linux 主机时， 你可以利用 ulimit -a 去察看一下，嘿嘿！他们能用的资源就减小很多了！ 更多的用法您可以自行参考一下该档案内的说明啊！ ^_^ (记得测试完毕要将资料改回来～否则....以后就麻烦了！)

Linux 系统上使用者的对谈与 mail 的使用：

谈了这么多的系统账号问题，总是该要谈一谈，那么如何针对系统上面的使用者进行查询吧？！ 想几个状态，如果你在 Linux 上面操作时，刚好有其它的使用者也登入主机，你想要跟他对谈，该如何是好？ 你想要知道某个账号的相关信息，该如何查阅？呼呼！底下我们就来聊一聊～

查询使用者： w, who, last, lastlog

如何查询一个使用者的相关数据呢？这还不简单，我们之前就提过了 w, who, finger 等指令了， 都可以让您了解到一个使用者的相关信息啦！那么想要知道使用者到底啥时候登入呢？ 最简单可以使用 last 检查啊！这个玩意儿我们也在 bash shell 那个章节提过了， 您可以自行前往参考啊！简单的很。不过， last 仅有列出这个月份的资料而已喔。

另外，如果您想要知道每个账号的最近登入的时间，则可以使用 lastlog 这个指令喔！ lastlog 会去读取 /var/log/lastlog 档案，结果将数据输出，如下表：

```
[root@linux ~]# lastlog
Username         Port     From             Latest
root             tty1                      Tue Aug 16 18:06:20 +0800 2005
bin                                        **Never logged in**
daemon                                     **Never logged in**
....以下省略.....
```

这样就能够知道每个账号的最近登入的时间啰～ ^_^

使用者对谈： talk, mesg, wall

那么我是否可以跟系统上面的使用者谈天说地呢？当然可以啦！ 利用 talk 这个指令即可！不过，talk 需要额外的启动一些网络服务， 对于目前的 Linux distribution 以及网络环境，嘿嘿！咱们还是不要玩这个东西啦～ 如果您确定想要玩这个玩意儿，那么请自行 man talk ， 同时考虑启动 ntalk 这个服务看看啰～

除了直接在线对谈 (talk) 之外，有没有其它讯息传送的功能啊？ 有啊！利用 write 是不错的方式啦！他可以直接将讯息传给接收者啰！ 举例来说，我们的 Linux 目前有 vbird 与 dmtsai 两个人在在线：

```
[vbird@linux ~]$ w
 16:50:39  up  1:58,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM      LOGIN@   IDLE   JCPU   PCPU  WHAT
vbird    pts/0    vbird     4:37pm   0.00s  0.06s  0.01s  w
dmtsai   pts/1    dmtsai    4:38pm   1:51   0.07s  0.01s  man write
```

我以 vbird 传送一个讯息给 dmtsai 时，可以这样做：

```
[vbird@linux ~]$ write dmtsai
Hi, How are you doing today....
Nothing.... just say hello to you!
<==这里按下 [ctrl]+d
```

此时，另一端的 dmtsai 在他的终端接口上就会出现如下所示：

```
[dmtsai@linux ~]$
Hi, How are you doing today....
Nothing.... just say hello to you!
EOF
```

怪怪～立刻会有讯息响应给 dmtsai 乀！不过......当时 dmtsai 正在查资料，哇！ 这些讯息会立刻打断 dmtsai 原本的工作喔！所以，如果 dmtsai 这个人不想要接受任何讯息， 直接下达这个动作：

```
[dmtsai@linux ~]$ mesg n
```

所以，当 vbird 再次下达 write 时，就会出现：

```
[vbird@linux ~]$ write dmtsai
write: dmtsai has messages disabled
```

呼呼！了解乎？而如果想要解开的话，再次下达『 mesg y 』就好啦！想要知道目前的 mesg 状态，直接下达『 mesg 』即可！瞭呼？

相对于 write 是仅针对一个使用者来传『简讯』，我们还可以『对所有系统上面的使用者传送简讯』哩～ 如何下达？呼呼！用 wall 即可啊！他的语法也是很简单的喔！

```
[root@linux ~]# wall "I will shutdown the linux server about 5m later.
> If you still have to login, please tell me.
> Or I will do it....."
```

那么除非您的 mesg 状态是 n ，否则，嘿嘿！就能够收到这个广播讯息啰！ ^_^

使用者邮件信箱： mail

使用 wall, write 毕竟要等到使用者在在线才能够进行，有没有其它方式来联络啊？ 不是说每个 Linux 主机上面的使用者都具有一个 mailbox 吗？ 我们可否寄信给使用者啊！呵呵！当然可以啊！我们可以寄、收 mailbox 内的信件呢！ 一般来说， mailbox 都会放置在 /var/spool/mail 里面，一个账号一个 mailbox (档案)。 举例来说，我的 dmtsai 就具有 /var/spool/mail/dmtsai 这个 mailbox 喔！

那么我该如何寄出信件呢？嗯！就直接使用 mail 这个指令即可！ 这个指令的用法很简单的，直接这样下达：『 mail username@localhost -s "邮件标题" 』即可！ 一般来说，如果是寄给本机上的使用者，基本上，连『 @localhost 』都不用写啦！ 举例来说，我以 vbird 寄信给 dmtsai ，信件标题是『 nice to meet you 』，则：

```
[vbird@linux ~]$ mail dmtsai -s "nice to meet you"
Hello, D.M. Tsai
Nice to meet you in the network.
You are so nice.  byebye!
.   <==这里很重要喔，结束时，最后一行输入小数点 . 即可！
Cc:  <==这里是所谓的『副本』，不需要寄给其它人，所以直接 [Enter]
[vbird@linux ~]$  <==出现提示字符，表示输入完毕了！
```

呼呼！如此一来，你就已经寄出一封信给 dmtsai 这位使用者啰，而且，该信件标题为： nice to meet you，信件内容就如同上面提到的。不过，你或许会觉得 mail 这个程序不好用～ 因为在信件编写的过程中，如果写错字而按下 Enter 进入次行，前一行的数据很难删除乀！ 那怎么办？没关系啦！我们使用数据流重导向啊！呵呵！利用那个小于的符号 (<) 就可以达到取代键盘输入的要求了。也就是说，你可以先用 vi 将信件内容编好， 然后再以 mail dmtsai -s "nice to meet you" < filename 来将档案内容传输即可。

例题：请将你的家目录下的环境变量文件（~/.bashrc）寄给自己！
答：

```
mail -s "bashrc file content" vbird < ~/.bashrc
```

刚刚上面提到的是关于『寄信』的问题，那么如果是要收信呢？呵呵！同样的使用 mail 啊！ 假设我以 dmtsai 的身份登入主机，然后输入 mail 后，会得到什么？

```
[dmtsai@linux ~]$ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/dmtsai": 1 message 1 new
>N  1 vbird@linux.site  Fri Sep  2 23:53  16/552   "nice to meet you"
&  <==这里可以输入很多的指令，如果要查阅，输入 ? 即可！
```

在 mail 当中的提示字符是 & 符号喔，别搞错了～输入 mail 之后，我可以看到我有一封信件， 这封信件的前面那个 > 代表目前处理的信件，而在大于符号的左边那个 N 代表该封信件尚未读过， 如果我想要知道这个 mail 内部的指令有哪些，可以在 & 之后输入『 ? 』，就可以看到如下的画面：

```
& ?
    Mail   Commands
t <message list>                type messages
n                               goto and type next message
e <message list>                edit messages
f <message list>                give head lines of messages
d <message list>                delete messages
s <message list> file           append messages to file
u <message list>                undelete messages
R <message list>                reply to message senders
r <message list>                reply to message senders and all recipients
pre <message list>              make messages go back to /usr/spool/mail
m <user list>                   mail to specific users
q                               quit, saving unresolved messages in mbox
x                               quit, do not remove system mailbox
h                               print out active message headers
!                               shell escape
cd [directory]                  chdir to directory or home if none given
```

<message list> 指的是每封邮件的左边那个数字啦！而几个比较常见的指令是：

指令	意义
h	将信件的标题列出来。如果想要查阅 40 封信件左右处的所有信件标头，可以输入『 h 40 』
d	删除啦～假设我要删除第 10 封信，可以『 d10 』，假如我想要删除 20-40 封信，可以『 d20-40 』，不过，这个动作要生效的话，必须要配合 q 这个指令才行(参考底下说明)！
s	将信件储存成为档案。举例来说，我要将第 5 封信件的内容存成 ~/mail.file 的话，可以：『 s 5 ~/mail.file 』喔！

x	或者输入 exit 都可以。这个是『不作任何动作离开 mail 程序』的意思。不论你刚刚删除了什么信件，或者读过什么，使用 exit 都会直接离开 mail，所以刚刚进行的删除与阅读工作都会无效。如果您只是查阅一下邮件而已的话，一般来说，建议使用这个离开啦！除非你真的要删除某些信件。
q	相对于 exit 是不动作离开， q 则会进行两项动作： 1. 将刚刚删除的信件移出 mailbox 之外； 2. 将刚刚有阅读过的信件存入 ~/mbox ，且移出 mailbox 之外。鸟哥通常不很喜欢使用 q 离开，因为，很容易忘记读过什么咚咚～导致信件给他移出 mailbox 说～

mail 这个指令内容还有很多可以玩的，不过，我们这里仅是介绍给您如何让一般身份使用者来使用 mail 而已，所以就介绍到此啰～ ^_^

手动增加使用者：

一般来说，我们不很建议大家使用手动的方式来新增使用者，为什么呢？ 因为使用者的建立涉及到 GID/UID 等权限的关系，而且，与档案/目录的权限也有关系， 使用 useradd 可以帮我们自动设定好 UID/GID 家目录以及家目录相关的权限设定， 但是，手动来增加的时候，有可能会忘东忘西，结果导致一些困扰的发生。

不过，要了解整个系统，最好还是手动来修改过比较好，至少我们的账号问题可以完全依照自己的意思去修订， 而不必迁就于系统的默认值啊！但是，还是要告诫一下朋友们，要手动设定账号时， 您必须要真的很了解自己在作什么，尤其是与权限有关的设定方面喔！ 好吧！底下就让我们来玩一玩啰～ ^_^

一些检查工具

既然要手动修改账号的相关设定档，那么一些检查群组、账号相关的指令就不可不知道啊～ 尤其是那个密码转换的 pwconv 及 pwuconv 这两个玩意～可重要的很呢！ 底下我们稍微介绍一下这些指令吧！

- pwck

pwck 这个指令在检查 /etc/passwd 这个账号设定文件内的信息，与实际的家目录是否存在等信息， 还可以比对 /etc/passwd /etc/shadow 的信息是否一致，另外， 如果 /etc/passwd 内的数据域位错误时，会提示使用者修订。 一般来说，我只是利用这个玩意儿来检查我的输入是否正确就是了。

```
[root@linux ~]# pwck
user adm: directory /var/adm does not exist
user news: directory /etc/news does not exist
user uucp: directory /var/spool/uucp does not exist
```

瞧！上面仅是告知我，这些账号并没有家目录，由于那些账号绝大部分都是系统账号， 确实也不需要家目录的，所以，那是『正常的错误！』呵呵！不理他。 ^_^。 相对应的群组检查可以使用 grpck 这个指令的啦！

- pwconv

这个指令主要的目的是在『将 /etc/passwd 内的账号与密码，移动到 /etc/shadow 当中！』早期的 Unix 系统当中并没有 /etc/shadow 呢，所以，使用者的登入密码早期是在 /etc/passwd 的第二栏，后来为了系统安全，才将密码数据移动到 /etc/shadow 内的。 使用 pwconv 后，可以：

- 比对 /etc/passwd 及 /etc/shadow ，若 /etc/passwd 内存在的账号并没有对应的 /etc/shadow 密码时，则 pwconv 会去 /etc/login.defs 取用相关的密码数据，并建立该账号的 /etc/shadow 数据；
- 若 /etc/passwd 内存在加密后的密码数据时，则 pwconv 会将该密码栏移动到 /etc/shadow 内，并将原本的 /etc/passwd 内相对应的密码栏变成 x ！

一般来说，如果您正常使用 useradd 增加使用者时，使用 pwconv 并不会有任何的动作，因为 /etc/passwd 与 /etc/shadow 并不会有上述两点问题啊！ ^_^。 不过，如果手动设定账号，这个 pwconv 就很重要啰！

- pwunconv

相对于 pwconv ， pwunconv 则是『将 /etc/shadow 内的密码栏数据写回 /etc/passwd 当中， 并且删除 /etc/shadow 档案。』这个指令说实在的，最好不要使用啦！ 因为他会将你的 /etc/shadow 删除喔！如果你忘记备份，又不会使用 pwconv 的话， 粉严重呢！

- chpasswd

chpasswd 是个挺有趣的指令，他可以『读入未加密前的密码，并且经过加密后， 将加密后的密码写入 /etc/shadow 当中。』这个指令很常被使用在大量建置账号的情况中喔！ 他可以由 Standard input 读入数据，每笔数据的格式是『 username:password 』。举例来说，我的系统当中有个使用者账号为 dmtsai ，我想要更新他的密码 (update) ， 假如他的密码是 abcdefg 的话，那么我可以这样做：

```
[root@linux ~]# echo "dmtsai:abcdefg" | chpasswd
```

神奇吧！这样就可以更新了呢！在预设的情况中， chpasswd 使用的是 DES 加密方法来加密， 我们可以使用 chpasswd -m 来使用 FC4 预设的 MD5 加密方法，不过， FC4 似乎怪怪的，我老是无法使用 -m 来达成这个指令。无论如何， 还是可以直接使用 chpasswd 来应用 DES 加密喔！ 使用 DES 方法加密后，在 /etc/shadow 的密码栏内，他的密码位数为 13 位， 瞭乎？？

特殊账号，如纯数字账号的建立

在我们了解了 UID/GID 与账号的关系之后，基本上，您应该了解了，为啥我们不建议使用纯数字的账号了！因为很多时候，系统会搞不清楚那组数字是『账号』还是『UID』，这不是很好啦～ 也因此，在早期某些版本底下，是没有办法使用数字来建立账号的。例如在 Red Hat 9 的环境中， 使用『 useradd 1234 』他会显示『 useradd: invalid user name '1234' 』呼呼！了解了吗？！ (不过，这个问题在 FC4 却不存在！因为 FC4 可以建立纯数字的账号说～)

不过，有的时候，长官的命令难为啊～有时还是得要建立这方面的账号的，那该如何是好？ 呵呵！当然可以手动来建立这样的账号啦！不过，为了系统安全起见，鸟哥还是不建议使用纯数字的账号的啦！ 因此，底下的范例当中，我们使用手动的方式来建立一个名为 normaluser 的账号， 而且这个账号属于 normalgroup 这个群组。OK！那么整个步骤该如何是好呢？ 由前面的说明来看，您应该了解了账号与群组是与 /etc/group, /etc/shadow, /etc/passwd, /etc/gshadow 有关，因此，整个动作是这样的：

1. 先建立所需要的群组 (vi /etc/group)；
2. 将 /etc/group 与 /etc/gshadow 同步化 (grpconv)；
3. 建立账号的各个属性 (vi /etc/passwd)；
4. 将 /etc/passwd 与 /etc/shadow 同步化 (pwconv)；
5. 建立该账号的密码 (passwd accountname)；
6. 建立使用者家目录 (cp -a /etc/skel /home/accountname)；
7. 更改使用者家目录的属性 (chown -R accountname.group /home/accountname)。

够简单的咯吧！让我们来玩一玩啰～

```
1. 建立群组 normalgroup ，假设 520 这个 GID 没有被使用！并且同步化 gshadow
[root@linux ~]# vi /etc/group
# 在最后一行加入底下这一行！
normalgroup:x:520:
[root@linux ~]# grpconv
[root@linux ~]# grep 'normalgroup' /etc/group /etc/gshadow
/etc/group:normalgroup:x:520:
/etc/gshadow:normalgroup:x::
# 简单！搞定群组啰！ ^_^

2. 建立 normaluser 这个账号，假设 UID 700 没被使用掉！
[root@linux ~]# vi /etc/passwd
# 在最后一行加入底下这一行！
normaluser:x:700:520::/home/normaluser:/bin/bash

3. 同步化密码，并且建立该使用者的密码
[root@linux ~]# pwconv
[root@linux ~]# grep 'normaluser' /etc/passwd /etc/shadow
/etc/passwd:normaluser:x:700:520::/home/normaluser:/bin/bash
/etc/shadow:normaluser:x:13030:0:99999:7:::
# 呵呵！没错没错！已经建立妥当啰～但是密码还不对～
[root@linux ~]# passwd normaluser
Changing password for user normaluser.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.

4. 建立使用者家目录，并且修订权限！
[root@linux ~]# cp -a /etc/skel /home/normaluser
[root@linux ~]# chown -R normaluser:normalgroup /home/normaluser
```

别怀疑！这样就搞定了一个账号的设定了！ 从此以后，你可以建立任何名称的账号啰～不过，还是不建议您设定一些很怪很怪的账号名称啦！

不开放终端机登入的账号 (ex>mail acccount)

刚刚我们上面建立的这个账号是『可以登入系统的账号』，如果想要建立一个不能登入系统的账号， 例如单纯使用邮件收发信件而已的账号，那么又该如何设定呢？很简单啦～ 你可以这样想：

- 因为不需要登入系统，所以建议 shell 字段给予 /sbin/nologin ;
- 因为不需要登入，所以家目录也可以先不建立。

也就是说，其实所有的步骤与刚刚上头提到的动作都一样，不过，少掉了很多与家目录有关的设定行为就是了。底下我假设我的系统里面有个叫做 mail 的群组 (/etc/group) ，他的 GID 是 12（以 FC4 为例），另外，这个使用者的账号为 popuser ，假设 UID 为 720 ，那么该如何建立呢？

```
1. 修改账号属性
[root@linux ~]# vi /etc/passwd
popuser:x:720:12::/home/popuser:/sbin/nologin

2. 密码同步，并且给予密码!
[root@linux ~]# pwconv
[root@linux ~]# passwd popuser
```

这样就又 OK 了～哇！真是太简单了杰克～....

那么又该如何删除这些账号呢？啊！还是建议利用 userdel 啦～简单～干脆又利落～ 如果想要暂时移除而已的话，那么利用 passwd -l 及 passwd -u 吧！ ^_^。 如果真的那么想要手动来移除这个账号的话，就这样做：

1. 先以 find / -user account 找出所有的账号档案，并将他删除；
2. 将 /etc/passwd 与 /etc/shadow 的相关资料删除；
3. 将 /etc/group 及 /etc/gshadow 相关资料删除；
4. 将 /home 底下关于该账号的目录删除；
5. 到 /var/spool/mail 以及 /var/spool/cron 里面将相关的使用者档案删除。

这样就手动删除啦～

一个大量建置账号的范例

不要怀疑，很多时候，我们都可能需要大量的建置账号的， 举例来说，学校要帮同学建立他们的账号，那就很可能需要啦～ 一般来说，建立账号要进行的前制工作很多，包括要建立账号名称与该账号的密码对应表～ 这个是最讨厌的啦～而且还要决定需要使用哪一个群组～呼呼～好讨厌的感觉那～

目前很多网站都有提供大量建立账号的工具，例如台南县网中心的卧龙大师：

http://linux.tnc.edu.tw/techdoc/howto/howtouse_cmpwd.htm

提供的好用的 cmpwd 程序， 不过，其实我们也可以利用简单的 script 来帮我们达成喔！例如底下这支程序， 他的执行结果与卧龙大师提供的程序差不多啦～ 但是因为我是直接以 useradd 来新增的， 所以，即使不了解 UID ，也是可以适用的啦～

整支程序的特色是：

- 预设不允许使用纯数字方式建立账号；
- 可加入年级来区分账号；
- 可设定账号的起始号码与账号数量；
- 有两种密码建立方式，可以与账号相同或程序自行以随机数建立密码文件。

执行方法也简单的要命～请自行参考的啦！不再多说～ 使用时请注意，不要在公家使用的主机上面进行测试，因为..... 这支程序会大量建立账号嘛！^_^

```
#!/bin/bash
#
# 这支程序主要在帮您建立大量的账号之用，
# 更多的使用方法请参考：
# http://linux.vbird.org/linux_basic/0410accountmanager.php#manual_amount
#
# 本程序为鸟哥自行开发，在 FC4 上使用没有问题，
# 但不保证绝不会发生错误！使用时，请自行负担风险～
#
# History:
# 2005/09/05    VBird   刚刚才写完，使用看看先～
PATH=/sbin:/usr/sbin:/bin:/usr/bin; export PATH
accountfile="user.passwd"

# 1. 进行账号相关的输入先！
read -p "账号开头代码 ( Input title name, ex> std )======> " username_start
read -p "账号层级或年级 ( Input degree, ex> 1 or enter )=> " username_degree
read -p "起始号码 ( Input start number, ex> 520 )========> " nu_start
read -p "账号数量 ( Input amount of users, ex> 100 )=====> " nu_amount
read -p "密码标准 1) 与账号相同 2)随机数自订 ==============> " pwm
if [ "$username_start" == "" ]; then
        echo "没有输入开头的代码，不给你执行哩！" ; exit 1
fi
testing1=`echo $nu_amount | grep '[^0-9]' `
testing2=`echo $nu_start  | grep '[^0-9]' `
if [ "$testing1" != "" ] || [ "$testing2" != "" ]; then
        echo "输入的号码不对啦！有非为数字的内容！" ; exit 1
fi
if [ "$pwm" != "1" ]; then
        pwm="2"
fi

# 2. 开始输出账号与密码档案！
[ -f "$accountfile" ] && mv $accountfile "$accountfile"`date +%Y%m%d`
```

```
nu_end=$(($nu_start+$nu_amount-1))
for (( i=$nu_start; i<=$nu_end; i++ ))
do
        account=$username_start$username_degree$i
        if [ "$pwm" == "1" ]; then
                password="$account"
        else
                password=""
                test_nu=0
                until [ "$test_nu" == "8" ]
                do
                        temp_nu=$(($RANDOM*50/32767+30))
                        until [ "$temp_nu" != "60" ]
                        do
                                temp_nu=$(($RANDOM*50/32767+30))
                        done
                        test_nu=$(($test_nu+1))
                        temp_ch=`printf "\x$temp_nu"`
                        password=$password$temp_ch
                done
        fi
        echo "$account":"$password" | tee -a "$accountfile"
done

# 3. 开始建立账号与密码!
        cat "$accountfile" | cut -d':' -f1 | xargs -n 1 useradd -m
        chpasswd < "$accountfile"
        pwconv
          echo "OK! 建立完成! "
```

这支程序可以在底下连结下载：

http://linux.vbird.org/download/index.php?action=download&fileid=70

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- root 的 UID 与 GID 是多少？而基于这个理由，我要让 test 这个账号具有 root 的权限，应该怎么作？

  root 的 UID 与 GID 均为 0 ，所以要让 test 变成 root 的权限，那么就将 /etc/passwd 里面，test 的 UID 与 GID 字段变成 0 即可！

- 假设我是一个系统管理员，我有一个用户最近不乖，所以我想暂时将他的账号停掉， 让他近期无法进行任何动作，等到未来他乖一点之后，我再将他的账号启用，请问：我可以怎么作比较好？？

由于这个账号是暂时失效的,所以不能使用 userdel 来删除,否则很麻烦!那么应该如何设定呢?再回去瞧一瞧 /etc/shadow 的架构，可以知道有这几个可使用的方法：

- o 将 /etc/passwd 的 shell 字段写成 /sbin/nologin ，即可让该账号暂时无法登入主机；
- o >将 /etc/shadow 内的密码字段，增加一个 * 号在最前面，这样该账号亦无法登入！
- o >将 /etc/shadow 的第八个字段关于账号取消日期的那个，设定小于目前日期的数字，那么他就无法登入系统了！

- 在设定密码的时候，是否可以随便设定呢？

  最好不要随便设定密码！最好可以仔细的参考一下本章内容提到的部分！

- 我在使用 useradd 的时候，新增的账号里面的 UID, GID 还有其它相关的密码控制，都是在哪几个档案里面设定的？

  在 /etc/login.defs 还有 /etc/default/useradd 里面规定好的！

- 我希望我在设定每个账号的时候(使用 useradd)，预设情况中，他们的家目录就含有一个名称为 www 的子目录，我应该怎么作比较好？

  由于使用 useradd 的时候,会自动以 /etc/skel 做为预设的家目录,所以,我可以在 /etc/skel 里面新增加一个名称为 www 的目录即可！

- pwconv 这个指令有什么功能呢？

  pwconf 可以让 passwd 里面的账号，设定一份密码到 /etc/shadow 当中！

- 简单说明系统账号与一般使用者账号的差别？

  一般而言,为了让系统能够顺利以较小的权限运作,系统会有很多账号， 例如 mail, bin, adm 等等。而为了确保这些账号能够在系统上面具有独一无二的权限， 一般来说 Linux 都会保留一些 UID 给系统使用。在 FC4 上面，小于 500 以下的账号 (UID) 即是所谓的 System account。

- 简单说明，为何 FC4 建立使用者时，他会主动的帮使用者建立一个群组，而不是使用 /etc/default/useradd 的设定？

  不同的 linux distributions 对于使用者 group 的建立机制并不相同。主要的机制分为：

  - o Public group schemes: 使用者将会直接给予一个系统指定的群组，一般来说即是 users ， 可以 SuSE Server 9 为代表；
  - o Private group schemes: 系统会建立一个与账号一样的群组名称！以 FC4 为例！

- 如何建立一个使用者名称 alex, 他所属群组为 alexgroup, 预计使用 csh, 他的全名为 "Alex Tsai"， 且他还得要加入 users 群组当中！

groupadd alexgroup
useradd -c "Alex Tsai" -g alexgroup -G users -m alex
务必先建立群组，才能够建立使用者喔!

- 由于种种因素，导致你的使用者家目录以后都需要被放置到 /account 这个目录下。 请问，我该如何作，可以让使用 useradd 时，预设的家目录就指向 /account ？

  最简单的方法,编辑 /etc/default/useradd ,将里头的 HOME=/home 改成 HOME=/account 即可。

- 我想要让 dmtsai 这个使用者，加入 vbird1, vbird2, vbird3 这三个群组，该如何动作？

  usermod -G vbird1,vbird2,vbird3 dmtsai

Linux 磁盘配额 (Quota)

最近更新日期：2005/09/06

磁盘配额（Quota）一直就是个很有用的东西！怎么说呢？举个例子来说明， 如果您曾经申请过网络的 mail 服务时，那么肯定就会明白什么是 20MB 的邮件空间、 30MB 的免费网页空间，好了，这个 20MB, 30MB 是怎样定义出来的呢？哈哈！没错，就是 quota 这个东西搞出来的！如果我们要限制使用者使用硬盘的容量使用大小，嗯！来这里看看就对了！

什么是 Quota

『 quota 』就字面上的意思来看，呵呵！就是有多少『限额』的意思啦！如果是用在零用钱上面， 就是类似『有多少零用钱一个月』的意思之类的。如果是在容量空间上面呢？以 Linux 来说，呵呵！就是有多少容量限制的意思。

在 Linux 系统中，由于是多人多任务的环境，所以会有多人共同使用一个硬盘空间的情况发生， 如果其中有少数几个使用者大量的占掉了硬盘空间的话，那势必压缩其它使用者的使用权力！ 因此管理员应该适当的开放硬盘的权限给使用者，以妥善的分配系统资源！避免有人抗议呀！ 举例来说，我们使用者的预设家目录都是在 /home 底下，如果 /home 是个独立的 partition ， 他大概有 10G 好了，而 /home 底下共有 30 个人，也就是说，每个使用者平均应该会有 333MB 的空间才对。 偏偏有个使用者在他的家目录底下塞了好多只影片，占掉了 8GB 的空间，想想看，是否造成其它正常使用者的不便呢？ 这个时候就得要靠 quota 的帮忙啰！

- Quota 的一般用途

quota 比较常使用的几个情况是：

- 针对 WWW server ，例如：每个人的网页空间的容量限制！
- 针对 mail server，例如：每个人的邮件空间限制。
- 针对 file server，例如：每个人最大的可用网络硬盘空间。

在 Linux 当中，使用来作为硬盘空间管理的就是所谓的 quota 这个咚咚啦！

- Quota 的使用限制

使用这个模块要有几个步骤，底下就分别说说吧！另外要特别注意的是，使用 quota 时有几个基本的限制需要谈一谈：

- 仅针对整个 partition：
  quota 实际在运作的时候，是针对『整个 partition』进行限制的， 例如：如果你的 /dev/hda5 是挂载在 /home 底下，那么在 /home 底下的所有目录都会受到限制！
- 核心必须支持 quota ：
  Linux 系统核心必须有支持 quota 这个模块才行：如果您是使用 FC4 的预设核心， 嘿嘿！那恭喜你了，你的系统已经预设有开放 quota 这个模块啰！如果您是自行编译核心的，那么请特别留意您是否已经『真的』开启了 quota 这个模块？否则底下的功夫将全部都视为『白工』。至于核心编译的过程我们会在未来进行说明呦！
- Quota 的记录文件：
  目前新版的 Linux distributions 如： FC4 与 SuSE server 9 等使用的是 Kernel 2.6.xx 的核心版本，这个核心版本支持新的 quota 模块，使用的预设档案(aquota.user, aquota.group)将不同于旧版本的 quota.user, quota.group ！ (多了一个 a 呦！)而由旧版本的 quota 可以藉由 convertquota 这个程序来转换呢！
- 只对一般身份使用者有效：
  这就有趣了！并不是所有在 Linux 上面的账号都可以设定 quota 呢，例如 root 就不能设定 quota ， 因为整个系统所有的数据几乎都是他的啊！ ^_^

- quota 这支程序对硬盘配额的限制项目：

quota 这支程序针对整个 partition 的限制项目主要分为底下几个部分：

- soft：
  这是最低限制容量的意思，使用者在宽限期间之内，他的容量可以超过 soft ，但必需要宽限时间之内将磁盘容量降低到 soft 的容量限制之下！
- hard：
  这是『绝对不能超过』的容量！跟 soft 相比的意思为何呢？通常 hard limit 会比 soft limit 为高，例如网络磁盘空间为 30 MB ，那么 hard limit 就设定为 30MB ，但是为了让使用者有一定的警戒心，所以当使用空间超过 25 MB 时，例如使用者使用了 27 MB 的空间时，那么系统就会警告使用者，让使用者可以在『宽限时间内』将他的档案量降低至 25 MB(亦即是 soft limit)之内！也就是说， soft 到 hard 之间的容量其实就是宽限的容量啦！可以达到针对使用者的『警示』作用！
- 宽限时间：
  那么宽限时间就可以很清楚的知道含意是什么了！也就是当您的使用者使用的空间超过了 soft limit ，却还没有到达 hard limit 时，那么在这个『宽限时间』之内， 就必需要请使用者将使用的磁盘容量降低到 soft limit 之下！而当使用者将磁盘容量使用情况超过 soft limit 时，『宽限时间』就会自动被启动，而在使用者将容量降低到 soft limit 之下，那么宽限时间就会自动的取消啰！

基本的 quota 指令：

在开始进行 quota 的实作之前，我们得来了解一下 quota 要使用的指令啰！基本上分为两种，一种是查询功能 (quota, quotacheck, quotastats, warnquota, repquota)，另一种则是编辑 quota 的内容 (edquota, setquota) 。底下我们来谈一谈这些基本的指令吧！

- /etc/mtab

怪了！不是说好要说明 quota 相关指令的吗？干嘛提这个档案系统 (filesystem) 实际挂载的记录文件？呵呵！要注意了～当我们使用 quota 的时候，基本上，系统会去搜寻：『 系统上具有 quota 参数的 partition 』 所以啰，当我们要使用 quota 的功能时，我们的 filesystem 必须要已经支持 quota 的旗标才行。一般来说，我们是以编辑 /etc/fstab 后，再重新挂载 filesystem 的方法来让系统的 filesystem 支持 quota 的！这个概念可是很重要的喔！ ^_^

- quota

```
[root@linux ~]# quota [-uvsl] [username]
[root@linux ~]# quota [-gvsl] [groupname]
参数：
-u  ：后面可以接 username ，表示显示出该使用者的 quota 限制值。若不接 username
      ，表示显示出执行者的 quota 限制值。
-g  ：后面可接 groupname ，表示显示出该群组的 quota 限制值。
-v  ：显示每个 filesystem 的 quota 值；
-s  ：可选择以 inode 或磁盘容量的限制值来显示；
-l  ：仅显示出目前本机上面的 filesystem 的 quota 值。
范例：

范例一：秀出目前 root 自己的 quota 限制值：
[root@linux ~]# quota -guvs

范例二：秀出 dmtsai 这个使用者的磁盘配额
[root@linux ~]# quota -vs -u dmtsai
# 注意一下这两个范例，如果您的系统上面尚未有任何的 quota 支持的 filesystem 时，
# 使用这两个范例时，『不会有任何信息列出来』啦！不要以为发生错误啰！
```

这个指令仅是使用来『显示(display)』目前某个群组或者某个使用者的 quota 限值！您可以使用来观察一下呦！<BR

- quotacheck

```
[root@linux ~]# quotacheck [-avug] [/mount_point]
参数：
-a  ：扫瞄所有在 /etc/mtab 内，含有 quota 支持的 filesystem，加上此参数后，
      /mount_point 可不必写，因为扫瞄所有的 filesystem 了嘛！
-u  ：针对使用者扫瞄档案与目录的使用情况，会建立 aquota.user
-g  ：针对群组扫瞄档案与目录的使用情况，会建立 aquota.group
-v  ：显示扫瞄过程的信息；
-M  ：『强制』进行 quotacheck 的扫瞄。
范例：

范例一：将所有的在 /etc/mtab 内，含有 quota 支持的 partition 进行扫瞄
[root@linux ~]# quotacheck -avug
```

```
quotacheck: Can't find filesystem to check or filesystem not mounted with
quota option.
# 不要紧张，这是正常的现象～因为您尚未启用 quota 的参数嘛！
# 关于 quota 参数的下达方法，我们会在稍后说明。如果正常的进行扫瞄，会像下面这样：
[root@linux ~]# quotacheck -avug
quotacheck: Scanning /dev/hdb1 [/disk2] done
quotacheck: Checked 3 directories and 4 files
[root@linux ~]# ll /disk2
total 32
-rw-------  1 root root  6144 Sep  5 14:56 aquota.group
-rw-------  1 root root  6144 Sep  5 14:56 aquota.user
drwx------  2 root root 16384 Jun 25 16:22 lost+found
# 第一次操作 quotacheck 可能会有一些错误讯息发生，那应该是正常的！
# 如果使用 ls -l 去查阅一下有 quota 支持的那个 mount point ，若有出现
# aquota.group 及 aquota.user ，那应该就是已经建立好了 quota 记录文件了！

范例二：强制扫瞄已挂载的 filesystem
[root@linux ~]# quotacheck -avug -m
# 有些时候，在某些 Linux distributions 上面，进行 quotacheck 时，
# 可能会出现如下的错误讯息：
# quotacheck: Cannot get quotafile name for /dev/hda3
# quotacheck: Cannot get quotafile name for /dev/hda3
# 果真如此的话，那么你可以如同上面一般，加上 -m 的参数来『强制』扫瞄。
# 也可以手动先建立记录文件，然后再扫瞄，如下所示：
[root@linux ~]# touch /disk2/aquota.user; touch /disk2/aquota.group
[root@linux ~]# quotacheck -avug
# 必须要注意的是，我这里是以 /disk2 作为一个测试的 mount point ，
# 您的挂载点不一定会跟鸟哥一样喔！
```

这这个指令主要的目的在扫瞄某一个磁盘的 quota 空间，他会针对该 partitions 进行扫瞄，并且，由于该磁盘若持续运作时，可能扫瞄的过程中， 档案可能会增减，造成 quota 扫瞄的错误发生，因此，当使用 quotacheck 时，该磁盘将『 自动被设定成为只读扇区 (read-only) 』 ；至于扫瞄完毕之后， 扫瞄所得的磁盘空间结果会写入该扇区最顶端。(例如：在鸟哥的例子中，扫瞄 /disk2 这个 /dev/hdb1 的扇区，如果是初次扫瞄，那么扫瞄完毕之后会产生 aquota.user 与 aquota.group ，会放置在 /disk2/aquota.user 与 /disk2/aquota.group 底下！而如果是建立 quota 后的扫瞄，那么就会更新这两个档案！) 另外， Linux 也特别强调 quota 在使用的时候，需要特别注意在 reboot 时，得先将 quota 关闭才好！

此外，由于新版的 Linux distribution 在 quota 的设计上似乎有点小问题，有时候无法完整的进行 quotacheck ，发生如同上表的情况，解决的方法就是主动手动的建立 quotafile 即可喔！例如上面的范例二所显示的。

- edquota

```
[root@linux ~]# edquota [-u username] [-g groupname]
[root@linux ~]# edquota -t  <==修改恕限时间
[root@linux ~]# edquota -p username_demo -u username
参数：
-u  ：后面接账号名称。可以进入 quota 的编辑画面 (vi) 去设定 username 的限制值；
-g  ：后面接群组名称。可以进入 quota 的编辑画面 (vi) 去设定 groupname 的限制值；
-t  ：可以修改恕限时间 (就是超过 quota 的 soft limit 值后，还能使用硬盘的宽限期限)
-p  ：复制范本。那个 username_demo 为已经存在并且已设定好 quota 的使用者，
      意义为『将 username_demo 这个人的 quota 限制值复制给 username 』！
范例：

范例一：设定 dmtsai 这个使用者的 quota 限制值
[root@linux ~]# edquota -u dmtsai
Disk quotas for user dmtsai (uid 501):
  Filesystem    blocks    soft    hard   inodes    soft    hard
  /dev/hdb1          0       0       0        0       0       0
# 进入编辑画面后，以 vi 的相关行为进行编辑喔！我们可以看到
# 被编辑的使用者是 dmtsai ，而底下共有七个字段，每个字段的意义我们将在
# 底下的说明继续介绍。而假设我们对于 dmtsai 的限制是 30MB 的话，那么：
Disk quotas for user dmtsai (uid 501):
  Filesystem    blocks    soft    hard   inodes    soft    hard
  /dev/hdb1          0   25000   30000        0       0       0
# 然后就可以储存后离开啰！

范例二：将 dmtsai 的 quota 限制值 (30MB) 复制给 vbird1 这个使用者
[root@linux ~]# edquota -p dmtsai -u vbird1

范例三：修订恕限时间
[root@linux ~]# edquota -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem             Block grace period     Inode grace period
  /dev/hdb1                     7days                  7days
# 预设的恕限时间是 7 天啦！你当然可以修订时间！
```

这个指令就是在编辑每一个『个人』或者是『群组』的 quota 数值！通常我们以 edquota -u username 或者是 edquota -g groupname 来编辑个人与群组的 quota 设定值。不过，或许您会觉得一个一个分配似乎很慢的样子！那么您也可以直接 copy 一个人的设定值给其它人，就如同上面第二个例子，利用已经建立好的 dmtsai 来建立 vbird1 这个人的 quota 限额！这个指令可是很重要的呦！ 另外，范例一当中出现的那七个字段代表的意义我们得要谈一谈啊：

- filesystem：
  代表这个 quota 是针对哪一个 partition 的意思。以范例一的情况来说，指的是 /dev/hdb1 啰！也就是 /disk2 那个目录底下的 quota 限制值啦！

- blocks：
  这个是目前使用者 dmtsai (uid 501) 在 /dev/hdb1 这个 filesystem (参考上面一个信息)，所耗掉的磁盘容量，也就是目前的使用掉的空间啦！单位是 Kbytes 喔！这个信息是 quota 程序自己计算出来的，所以请不要修改他！
- soft 与 hard ：
  这个是目前的 dmtsai 使用者在这个 filesystem 之内的 quota 限制值！至于 soft 与 hard 的意思就如同 前一节最后面提的那个意思啦！soft 代表的是一个『警告』限值， hard 则是一个『不可超过的限值』， soft 与 hard 中间的差值则为宽限的数值。而当 soft 与 hard 数值为 0 的时候，表示『没有限制』的意思！而数值的单位仍是 Kbytes 喔！
- inodes：
  是目前使用掉 inode 的状态，也是 quota 自己计算出来而得到的，所以不要去变更他。一般而言， inode 不容易控制，所以您可以不必去限制 inode 呢！

- quotaon

```
[root@linux ~]# quotaon [-avug]
[root@linux ~]# quotaon [-vug] [/mount_point]
参数：
-u  ：针对使用者启动 quota (aquota.user)
-g  ：针对群组启动 quota (aquota.group)
-v  ：显示启动过程的相关讯息；
-a  ：根据 /etc/mtab 内的 filesystem 设定启动有关的 quota ，若不加 -a 的话，
      则后面就需要加上特定的那个 filesystem 喔！
范例：

范例一：启动所有的具有 quota 的 filesystem
[root@linux ~]# quotaon -auvg
/dev/hdb1 [/disk2]: group quotas turned on
/dev/hdb1 [/disk2]: user quotas turned on

范例二：仅启动 /disk2 里面的 user quota 设定值：
[root@linux ~]# quotaon -uv /disk2
```

这个指令是在启动 quota 的！不过，由于这个指令是启动 aquota.group 与 aquota.user 的，所以您就必须要先完成 qutoacheck 的工作了！然后简单的下达 quotaon -a 即可启动！

- quotaoff

```
[root@linux ~]# quotaoff [-a]
[root@linux ~]# quotaoff [-ug] [/mount_point]
参数：
-a  ：全部的 filesystem 的 quota 都关闭 (根据 /etc/mtab)
-u  ：仅针对后面接的那个 /mount_point 关闭 user quota
-g  ：仅针对后面接的那个 /mount_point 关闭 group quota
```

```
范例：

范例一：
[root@linux ~]# quotaoff -a
```
这个指令就是关闭了 quota 的限制啦！

实作 Quota

Quota 使用的方向很广啦，不过，他一般的用途大概有这些：

- 限制某一群组所能使用的最大磁盘配额（使用群组限制）：
  你可以将你的主机上的使用者分门别类，有点像是目前很流行的付费与免付费会员制的情况， 你比较喜好的那一群的使用配额就可以给高一些！呵呵！ ^_^...
- 限制某一使用者的最大磁盘配额（使用使用者限制）：
  在限制了群组之后，您也可以再继续针对个人来进行限制，使得同一群组之下还可以有更公平的分配！
- 以 Link 的方式，来使邮件可以作为限制的配额（更改 /var/spool/mail 这个路径)：
  如果是分为付费与免付费会员的『邮件主机系统』，是否需要重新再规划一个硬盘呢？ 也不需要啦！直接使用 Link 的方式指向 /home （或者其它已经做好的 quota 磁盘）就可以啦！ 这通常是用在原本规划不好，但是却又不想要更动原有主机架构的情况中啊！

那么 Quota 从开始准备 filesystem 的支持到整个设定结束的主要的步骤大概是：

1. 设定 partition 的 filesystem 支持 quota 参数：
   由于 quota 必须要让 partition 上面的 filesystem 支持才行，一般来说， 支持度最好的是 ext2/ext3 ，其它的 filesystem 类型鸟哥我是没有试过啦！ 启动 filesystem 支持 quota 最简单就是编辑 /etc/fstab ，使得准备要开放的 quota 磁盘可以支持 quota 啰；
2. 建立 quota 记录文件：
   刚刚前面讲过，整个 quota 进行磁盘限制值记录的档案是 aquota.user/aquota.group， 要建立这两个档案就必须要先利用 quotacheck 扫瞄才行喔！所以啰，接下来的步骤就是： 使用 quotacheck 来扫瞄一下我们要使用的磁盘啰；
3. 编辑 quota 限制值数据：
   再来就是使用 edquota 来编辑每个使用者或群组的可使用空间啰；
4. 重新扫瞄与启动 quota ：
   设定好 quota 之后，建议可以再进行一次 quotacheck ，然后再以 quotaon 来启动吧！

整个 quota 设定的步骤就只是这样而已，简单吧！我们底下就直接来用一个范例介绍一下整个流程， 好让您更清楚的了解到整个步骤喔！我范例是这样的：

1. 鸟哥的这部 Linux 主机里面主要针对 quser1 及 quser2 两个使用者来进行磁盘配额， 且这两个使用者都是挂在 qgroup 群组里面的喔。
2. 每个使用者总共有 50MB 的磁盘空间（不考虑 inode）限制！并且 soft limit 为 45 MB；

3. 而宽限时间设定为 1 天，也就是说，这两个人可以突破 45MB 的限制， 但是在一天之内必须要将多余的档案砍掉，否则将无法使用剩下的空间（也就是说，这个账号大概就不能进行档案新增的工作了）；
4. gquota 这个群组考虑最大限额，所以设定为 90 MB 好了！

多说无用，我们就实际来进行啰！

---

1. 准备好测试的环境，使用者与群组的建立：

   这两个账号应该是不存在我们的系统的，所以，赶紧将他设定上去吧！

```
[root@linux ~]# groupadd qgroup
[root@linux ~]# useradd -m -g qgroup quser1
[root@linux ~]# useradd -m -g qgroup quser2
[root@linux ~]# passwd quser1
[root@linux ~]# passwd quser2
```

---

2. 建立好 filesystem 的 quota 支持：

   由于 quota 较完整的支持是需要在 ext2/ext3 的 Linux 延伸格式档案才可以启动， 所以建议你就必须要将准备开启 quota 的磁盘启动参数，写进入 quota 的磁盘设定才行 (/etc/fstab)！以鸟哥的例子而言，我想要在 /disk2 底下进行 quota 的限制 quser1, quser2 这两个人！这是因为我的 /disk2 是一个独立的扇区，这可以使用 df 来查询。 此外，必需要特别留意的是，最好不要以根目录亦即是 / 进行 quota 啦!否则容易有些问题呢！ 另外，不要针对 root 做 quota 喔！反正做了也没用！

```
[root@linux ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1              5952252   3193292   2451720  57% /
/dev/hdb1             28267608     77904  26730604   1% /disk2
/dev/hda5              9492644    227252   8775412   3% /disk1
```

   嗯！我的 /disk2 是独立的 partition ，并且他的装置名为 /dev/hdb1 ，好了！那么我就必须要启动 /disk2 这个 /dev/hdb1 的 quota 档案格式，好了！那么由于档案格式的设定是写在 /etc/fstab 里头，所以我们以 vi 来编辑他吧！只要在 /etc/fstab 里头增加了 usrquota, grpquota 就可以啦！ (注：请特别留意，这两个项目请『务必』不要写错了！请在写入 /etc/fstab 之前好好的再次检查，因为写错之后，很有可能造成系统无法开机，虽然机率不高，但是有可能！)

```
[root@linux ~]# vi /etc/fstab
LABEL=/         /           ext3    defaults                        1 1
LABEL=/disk1    /disk1      ext3    defaults                        1 2
LABEL=/disk2    /disk2      ext3    defaults,usrquota,grpquota      1 2
/dev/hda3       swap        swap    defaults                        0 0
```

注意到我们所需要设定的那个 /disk2 的那一行，在第四字段多了 usrquota,grpquota 注意，在『 defaults,usrquota,grpquota 』之间都没有空格!

这样就算加入了 quota 的磁盘格式了!不过，由于真正的 quota 在读取的时候是读取 /etc/mtab 这个档案的，偏偏这一个档案需要重新开机之后才能够以 /etc/fstab 的新数据进行改写! 所以这个时候你可以选择：

1. 重新开机 (reboot) ;
2. 重新 remount filesystem 来驱动设定值!

我是不太喜欢重新开机的人啦! 所以我就这么做：

```
[root@linux ~]# umount /dev/hdb1
[root@linux ~]# mount -a
[root@linux ~]# grep '/disk2' /etc/mtab
/dev/hdb1 /disk2 ext3 rw,usrquota,grpquota 0 0

# 事实上，也可以利用 mount 的 remount 功能!
[root@linux ~]# mount -o remount /disk2
```

嘿嘿嘿嘿! 这样我们就已经成功的将 filesystem 的 quota 功能加入啰! 另外，鸟哥这里是以 ext3 这个磁盘格式来测试 quota 的呦!

---

3. 扫瞄磁盘的使用者使用状况，并产生重要的 aquota.group 与 aquota.user:

接着下来就是要来扫瞄一下我们所需要的磁盘到底有没有多余的空间可以让我们来设定 quota 呢? 并且将扫瞄的结果输出到这个磁盘的最顶层去 (也就是 /disk2 底下) 这个时候就需要 quotacheck 这个指令的帮忙了!使用 quotacheck 就可以轻易的将所需要的数据给他输出了! 并且在 /disk2 底下会产生 aquota.group 与 aquota.user 这两个档案!

```
[root@linux ~]# quotacheck -avug
```

```
quotacheck: Scanning /dev/hdb1 [/disk2] done
quotacheck: Checked 3 directories and 4 files
[root@linux ~]# ll /disk2
-rw-------  1 root root  6144 Sep  6 11:44 aquota.group
-rw-------  1 root root  6144 Sep  6 11:44 aquota.user
```

使用 quotacheck 就可以轻易的将所需要的数据给他输出了！但是很奇怪的是，在某些 Linux 版本中，我不能够以 aquota.user(group) 来启动我的 quota ，这有可能是因为旧版 quota 的关系， 所以我就另外做了一个 link 档案来欺骗 quota 啰：

```
[root@linux ~]# cd /disk2
[root@linux ~]# ln -s aquota.user quota.user
[root@linux ~]# ln -s aquota.group quota.group
# 除非您的 Linux distributions 是比较旧的版本，否则不会有这个问题，
# 所以，这个动作你不必进行的！
```

---

4. 启动 quota 的限额：

再来就是要启动 quota 啦！启动的方式也是很简单的！就是使用 quotaon -av 即可：

```
[root@linux ~]# quotaon -avug
/dev/hdb1 [/disk2]: group quotas turned on
/dev/hdb1 [/disk2]: user quotas turned on
```

注意：要看到上面有个 turned on 的出现，才是真的成功了！

---

5. 编辑使用者的可使用空间：

由于我们有两个使用者要设定，先来设定 quser1 好了，使用 edquota 就对了：

```
[root@linux ~]# edquota -u quser1
Disk quotas for user quser1 (uid 502):
  Filesystem    blocks    soft    hard   inodes   soft   hard
  /dev/hdb1          0   45000   50000        0      0      0
```

再次强调的是，因为我的 /disk2 里面并没有任何数据存在，所以，在上面这个表格当中，blocks 与 inodes 才会都是 0 ，如果您是使用 /home 来进行 quota 设定的， 那么 blocks/inodes 肯定不会是 0 ，这里要特别留意的。好了，上面特殊字体的部分就是我们的设定了， 分别是 45000 及 50000 ，那个单位是 KBytes 啦，转成 MBytes 应该是要除以 1024 才对， 不过，简单算一下就好了，不要太介意喔！^_^。然后将 quser1 的设定直接复制给 quser2 吧！

```
[root@linux ~]# edquota -p quser1 quser2
```

接下来要来设定宽限时间，还是使用 edquota ！

```
[root@linux ~]# edquota -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem             Block grace period     Inode grace period
  /dev/hdb1                     1days                    7days
```

将时间改为 1 天（原本是 7days 改成 1days ），好了！查询一下是否真的有设定进去呢？使用 quota -v 来查询：

```
[root@linux ~]# quota -vu quser1 quser2
Disk quotas for user quser1 (uid 502):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/hdb1       0   45000   50000               0       0       0
Disk quotas for user quser2 (uid 503):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/hdb1       0   45000   50000               0       0       0
```

特别注意到，由于我们的使用者尚未超过 45 MB，所以 grace （ 宽限时间 ） 就不会出现啦！这样很够清楚了吧？！

---

6. 编辑群组可使用的空间：

```
[root@linux ~]# edquota -g qgroup
Disk quotas for group qgroup (gid 502):
  Filesystem     blocks     soft     hard   inodes    soft     hard
  /dev/hdb1           0    80000    90000        0       0        0

[root@linux ~]# quota -vg qgroup
Disk quotas for group qgroup (gid 502):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
```

```
    /dev/hdb1        0    80000    90000                  0        0        0
```

这样就设定好了 group 的 quota 啰！同样的，因为整个群组的总使用量还没有到达 80000 KBytes， 当然那个 grace 就不会有任何信息显示了！但这个地方倒是有很多朋友来信问到一个小问题， 那就是『为什么我两个使用者 quser1, quser2 的设定值在 soft 与 hard 分别是 45/50MB ，但为何你的 group 总量 (hard) 设定仅有 90MB 呢？』，也就是说， 当我的某个使用者用了 50MB 的量，那另一个不就最多可以使用到 40MB 而已？原因何在啊？

这么说好了，如果是小型的系统，由于使用者并不是很多，我们可以针对每个人来进行 quota 的设定值，所以，当然针对 users 来进行设定即可，不需要额外的设定 group 的 quota 设定啦。

但如果换个角度来思考，假设您所处的公司人员比较多且分工较细，因此， 我们可能无法真正了解每个使用者的需求，此时，针对每个使用者来设定可能就比较麻烦一点。 那么我们反过来说，可以针对每个部门 (group) 来进行 quota 的设定， 因为部门的需求直接跟部门的负责人询问就好了，比较容易，而该部门的使用者 quota 设定当然可以高一点， 因为，可能某些使用者有较为独特的需求啊！反正只要符合 group 的限制即可， 该部门如果超过整个 group quota 限制值，呵呵！让他们自己去处理即可！ ^_^

---

7. 设定开机时启动 quota：

这个部分就不需要担心了，因为 FC4 与 Red Hat 系列的开机 script (/etc/rc.d/rc.sysinit) 已经将 quota 的侦测写入在里头，因此，在预设的情况下， quota 是会主动的被启动的。 不过，如果你想要手动的强制 quota 在开机启动一遍，那么可以使用 vi 去编辑 /etc/rc.d/rc.local ，在里面加入一行 (直接加在最后一行即可)：

```
[root@linux ~]# vi /etc/rc.d/rc.local
/sbin/quotaon -avug
```

如果要关闭 qutoa 就是用 quotaoff 吧！没错！这样就将 quota 设定完毕了！很简单吧！！ (如果是 SuSE Server 9 的话，可能就要去修改 /etc/init.d/boot.local 这个档案啰！)

---

8. 利用 repquota 显示更完整的 quota 结果报告：

事实上，除了 quota 可以用来观察使用者与群组使用的 quota 限制值之外，其实， 我们还可以使用更详细的 quota 报告指令，就是 repquota 这个指令呢！他的基本用法是这样的：

```
[root@linux ~]# repquota -a [-vug]
参数：
```

```
-a  ：直接到 /etc/mtab 搜寻具有 quota 标志的 filesystem ，并报告 quota 的结果；
-v  ：输出所有的 quota 结果，而非仅下达指令者自己的 quota 限值；
-u  ：显示出使用者的 quota 限值 (这是默认值)；
-g  ：显示出个别群组的 quota 限值。
范例：

范例一：查阅系统内所有的具有 quota 的 filesystem 的限值状态：
[root@linux ~]# repquota -av
*** Report for user quotas on device /dev/hdb1
Block grace time: 24:00; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   77888       0       0              4     0     0
quser1    --       0   45000   50000              0     0     0
quser2    --       0   45000   50000              0     0     0

Statistics:          <==这是所谓的系统相关信息，用 -v 才会显示
Total blocks: 7
Data blocks: 1
Entries: 3
Used average: 3.000000

范例二：仅列出 user 与 group 的 quota 限值：
[root@linux ~]# repquota -aug
*** Report for user quotas on device /dev/hdb1
Block grace time: 24:00; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   77888       0       0              4     0     0



*** Report for group quotas on device /dev/hdb1
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
Group           used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   77888       0       0              4     0     0
```

根据这些信息，您就可以知道目前的限制情况啰！ ^_^

不更动既有系统的 quota 实例

好了，我们前面 账号管理 的部分曾经提到 e-mail 这个东西嘛！如果我们要设定一个对外开启的邮件主机的时候， 那么最好对于邮件空间有点限制比较好，免得如同上面提到的一些问题一样，造成使用者的使用权不一！所以说，使用 quota 确实是一个好建议！这个时候该怎么办呢？

什么怎么办？嗯！是这样的，由于 quota 『只能针对整个 partition 进行整体的磁盘配额， 无法针对某个目录进行磁盘配额！』针对这个观念，我们不难发现，『 (1)将邮件存在个人的家目录与 (2)将邮件统一放在 /home 下的一个共享目录 』是一样的！为什么呢？这是因为 quota 针对的是整个磁盘呀！呵呵！所以啰，您必须先确定『您的 /home 是一个独立的 partition 』才行！

不过，很可惜的是，当初我们进行 Linux 主机安装时，如果忘记将 /home 独立成一个 partition 时， 那该怎么办？是否需要将 /home 进行重新分割与挂载？还有，如果也忘记将 /var/spool/mail 这个 mailbox 放置的目录独立出来，又该如何是好啊？举个简单的例子来说，在鸟哥上面的那个实作当中， 你会发现，我的 partition 仅有 /, /disk1, /disk2 ，那我的所有使用者都在 /home 里面， 邮件在 /var/spool/mail 底下，真要命喔！怎么办啊？

其实没有怎么难啦！既然 quota 是针对整个 partition 来进行限制，那我又已经将 /disk2 做好 quota 了， 那么我只要：

1. 将 /home 这整个目录搬移到 /disk2 底下；
2. 利用 ln -s /disk2/home /home 来建立连结数据；
3. 将 /var/spool/mail 整个搬移到 /disk2 底下；
4. 利用 ln -s /disk2/mail /var/spool/mail 来建立连结数据。

只要这样的一个小步骤，嘿嘿！您家主机的邮件就有一定的限额啰！当然啰！您也可以依据不同的使用者与群组来设定 quota 然后同样的以上面的方式来进行 link 的动作！嘿嘿嘿！就有不同的限额针对不同的使用者提出啰！很方便吧！！ ^_^

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 如果我有一个 Linux 主机，未来想要对外提供 mail 与 WWW 的服务，同时预计提供个人网页空间的服务，然而我希望 mail 提供 30 MB 而 WWW 则提供 20MB 的空间，那么我应该如何规划我的主机？

  在 quota 的限制中，由于他限制的是整个 partition 呢！所以既然要分为两个服务来限制，就需要设定成两个 partition 了！这个案例当中是以 Linux 为新架设的角度来看，所以我们的规划就较为简单！假设我的硬盘为 30GB 的硬盘，那么我可以这样设定：

      / 256 MB
      Swap 2 * RAM

/usr 3~5 GB
/backup 5GB

其它的空间平均分给

/home
/var/spool/mail

这样就可以啦！然后安装完成之后，套用 quota 的设定，即可做好限制啰！很是方便的！

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 如果我有一个 Linux 主机，未来想要对外提供 mail 与 WWW 的服务，同时预计提供个人网页空间的服务，然而我希望 mail 提供 30 MB 而 WWW 则提供 20MB 的空间，那么我应该如何规划我的主机？

  在 quota 的限制中，由于他限制的是整个 partition 呢！所以既然要分为两个服务来限制，就需要设定成两个 partition 了！这个案例当中是以 Linux 为新架设的角度来看，所以我们的规划就较为简单！假设我的硬盘为 30GB 的硬盘，那么我可以这样设定：

  / 256 MB
  Swap 2 * RAM
  /usr 3~5 GB
  /backup 5GB

  其它的空间平均分给

  /home
  /var/spool/mail

  这样就可以啦！然后安装完成之后，套用 quota 的设定，即可做好限制啰！很是方便的！

例行性命令的建立

最近更新日期：2005/09/07

不论什么时候，建立系统可以自动的执行功能都是必须的！您了解目前系统在预设的情况中， 每天、每小时、每个月都在做些什么事吗？！您了解『循环的工作』与『仅进行一次的工作』有什么不同吗？ 还有还有，如果你想要每年的老婆的生日前一天就发出一封信件提醒自己不要忘记； 又，如果是与初次见面的朋友的约会，又该如何设定啊？看看这一章先！

什么是例行性命令？

每个人或多或少都有一些约会或者是工作，有的工作是例行性的，例如每年一次的加薪、 每个月一次的工作报告、每周一次的午餐会报、每天需要的打卡等等； 有的工作则是临时发生的，例如刚好总公司有高官来访，需要你准备演讲器材等等！ 用在生活上面，例如每年的爱人的生日、每天的起床时间等等、还有突发性的计算机大降价 (阿～鸟哥等好久了～～) 等等啰。

上面这些工作都可以称为例行性命令，而这些工作 Linux 也都可以帮您提醒，例如：每一天早上 8:00 钟要服务器连接上音响，并启动音乐来唤你起床；而中午 12:00 希望 Linux 可以发一封信到你的邮件信箱，提醒你可以去吃午餐了； 另外，在每年的你爱人的生日的前一天，先发封信提醒你，以免忘记这么重要的一天。

Linux 工作排程的种类： at, cron

从上面的说明当中可以很清楚的发现两种工作排程的方式：

- 一种是例行性的，就是每隔一定的周期要来办的事项；
- 一种是突发性的，就是这次做完以后就没有的那一种（计算机大降价....）

那么在 Linux 底下如何达到这两个功能呢？呵呵！那就得使用 at 与 crontab 这两个好东西啰！

- at ： 这个工作仅执行一次就从 Linux 系统中的排程中取消；
- cron ： 这个工作将持续例行性的作下去！

底下我们先来谈一谈 Linux 的系统到底在做什么事情，怎么有若干多的工作排程在工作呢？然后再回来谈一谈 at 与 crontab 这两个好东西！

系统上常见的例行性命令有哪些？

好了，那么服务器自己有什么例行性命令要来作呀！？Linux 的工作可多着呢！ 由前面提到的几篇文章中，我们知道 Linux 本身在背景下的工作可是很多的，尤其是开放网络联机的情况下，建立与取消联机、MySQL 数据库的实时更新、以及一些例行的系统指令，例如释放内存的工作等等。 由于例行的工作非常的多，实在不可能每天都要管理员来手动输入吧！ 所以才会建立这个工作排程的需求的！基本预设的工作有底下这些：

- 进行登录文件的数据轮替 (log rotate)：
  这个步骤重要了！尤其是在 log file 的选项当中！由于登录档案会越来越大， 所以需要适时适量的将登录档备份，并以新开的档案来进行记录，这样效率会比较好，因此就需要使用 log rotate 啦！系统预设的重要工作之一；
- rpm 数据库的建立：
  虽然 RPM 数据库会在你以 RPM 安装之后即更新到 RPM 数据库当中去，但是难保会有漏网之鱼，所以系统也会设定每隔依段时间自动的搜集系统上面的 RPM 数据库来建置一番；
- 建立 locate 的数据库：
  是否还记得为何使用 locate 这个指令时，搜寻速度超快！那是因为 Linux 系统上将档案与路径都记录在数据库里面了！所以使用 locate 的时候，嘿嘿！直接指向数据库去 (/var/lib/slocate/slocate.db) ，偏偏麻烦的是这个档案的更新是每天一次！所以当你今天更新的档案，使用 locate 反而可能会找不到....
- 进行程序的分析：
  每隔依段时间会进行程序的分析，如果发现有僵尸程序的时候，就会将他删去！ 以保持内存的工作能力！
- 登录档视察：
  这个东西是在 Red Hat 7.1 以后才出现的东西，后来太好用了，所以被拿到旧版的 Red Hat 里面去使用！基本上就是分析登录档啦！然后据以解析有问题的纪录文件， 以维护主机的安全性！这部份不才小弟也自己写了一个简易型的分析档案，觉得更好用就是了！
- 指纹数据库的比对：
  基本上就是 tripwire 这个套件啦!可以用来分析最近被更动过的档案内容！蛮不错的一个程序！有空也来玩玩看。

Linux 预设的例行工作至少就有这些了，再加上您努力的为 Linux 进行工作排程的设计， 嘿嘿！每天的工作量可是相当的大的呢！

仅进行一次的工作排程： at

好了，如同上面提到的，工作排程有所谓的例行性的，也有单一执行一次的，我们先来谈一谈仅执行一次的工作。 要使用这种工作排程时，我们的 Linux 系统上面必须要有负责这个排程的服务，那就是 atd 这个玩意儿。 可惜的是，目前挺多新的 Linux distributions 似乎预设不把他打开了，所以呢，我们必须要先手动将他启用才行。 启用的方法很简单，就是这样：

```
[root@linux ~]# /etc/init.d/atd restart
Stopping atd:                          [FAILED]
Starting atd:                          [  OK  ]

# 再设定一下开机时就启动！
[root@linux ~]# chkconfig --level 35 atd on
```

看到那个『 OK 』的字样就好啰～关于服务的启动，我们会在后续再加以介绍，如果您真的有兴趣， 那么可以自行到 /etc/init.d/atd 这个 shell script 内去瞧一瞧先！ ^_^。 至于那个 chkconfig ，呵呵！您也可以使用 man 先查阅一下啊！我们未来再介绍啦！

- at 的工作

既然是工作排程，那么自然要有写入工作的纪录文件啰！没错啦！ 我们可以使用 at 这个指令来帮忙写入工作纪录文件，工作纪录文件预设的放置目录在 /var/spool/at 底下，在写入 at 记录文件后，该工作便进入排程当中并等待执行。 当然啦，要让 /var/spool/at 目录底下的工作被实际运作，必须要启动我们上面提到的 atd 那个服务啦！

不过，并不是所有的人都可以进行 at 工作排程喔！为什么？因为安全的理由啊～ 很多主机被所谓的绑架后，最常发现的就是他们的系统当中有很多的怪客程序 (cracker program) 被写入例行性命令的排程当中了，所以，那些可恶的程序就可能定时或不定时的在你的系统当中工作， 呵呵！所以啰，除非是您认可的账号，否则先不要让他们使用 at 吧！ 此外，我们可以利用 /etc/at.allow 与 /etc/at.deny 这两个档案来进行 at 的使用限制呢！ 加上这两个档案后， at 的工作情况其实是这样的：

1. 先找寻 /etc/at.allow 这个档案，写在这个档案中的使用者才能使用 at ，没有在这个档案中的使用者则不能使用 at (即使没有写在 at.deny 当中)；
2. 如果没有 /etc/at.allow 就寻找 /etc/at.deny 这个档案，若写在这个 at.deny 的使用者则不能使用 at ，而没有在这个 at.deny 档案中的使用者，就可以使用 at 咯；
3. 如果两个档案都不存在，那么只有 root 可以使用 at 这个指令。

上面的情况说明了，其实我们只要有 at.deny 这个档案存在就好了，因为我们假设系统内的账号都是懂得操作的使用者， 因此，预设让他们可以任意使用 at 这个好用的东西！这也是系统的默认值。我们的 FC4 预设也是只有 /etc/at.deny 存在，而且该档案内并未有任何账号数据！这表示任何人均可使用 at 啦！不过，万一你不希望有某些使用者使用 at 的话，将那个使用者的账号写入 /etc/at.deny 即可！ 一个账号写一行。

- 开始使用 at 啰：

好了，让我们来谈一谈 at 这个玩意儿的语法吧！

```
[root@linux ~]# at [-m] TIME
参数：
-m  ：当 at 的工作完成后，以 email 的方式通知使用者该工作已完成。
TIME：时间格式，这里可以定义出『什么时候要进行 at 这项工作』的时间，格式有：
  HH:MM                                  ex> 04:00
```

```
          在今日的 HH:MM 时刻进行，若该时刻已超过，则明天的 HH:MM 进行此工作。
  HH:MM YYYY-MM-DD            ex> 04:00 2005-12-03
          强制规定在某年某月的某一天的特殊时刻进行该工作！
  HH:MM[am|pm] [Month] [Date]  ex> 04pm December 3
          也是一样，强制在某年某月某日的某时刻进行！
  HH:MM[am|pm] + number [minutes|hours|days|weeks]
          ex> now + 5 minutes  ex> 04pm + 3 days
          就是说，在某个时间点『再加几个时间后』才进行。
范例：

范例一：再过五分钟后，将 /root/.bashrc 寄给 dmtsai 这个使用者
[root@linux ~]# at now + 5 minutes
at> /bin/mail dmtsai -s "testing at job" < /root/.bashrc
at> <EOT>   <==这里输入 [ctrl] + d 就会出现 <EOF> 的字样！代表结束！
job 8 at 2005-09-07 10:47
# 上面这行信息在说明，第 8 个 at 工作将在 2005/09/07 的 10:47 进行！

范例二：由于机房预计于 2005/09/16 停电，我想要在 2005/09/15 23:00 关机？
[root@linux ~]# at 23:00 2005-09-15
at> /bin/sync
at> /bin/sync
at> /sbin/shutdown -h now
at> <EOT>
job 10 at 2005-09-15 23:00
# 您瞧瞧！ at 还可以在一个工作内输入多个指令呢！不错吧！
```

事实上，当我们使用 at 时，at 会给使用者一个 bash shell 让使用者下达工作指令， 此时，建议你最好使用绝对路径来下达你的指令，比较不会有问题喔！那我们知道每个指令都可能会有 standard output/standard error 啊，这些可能会输出到屏幕上面的信息会跑去哪里？ 呵呵！这些本来应该在屏幕上面出现的信息通通会以 email 的方式传送到使用者的 mailbox 里面去！ 而预设如果没有 stdout/stderr 时，就不会有任何讯息传送给使用者了。但你可以使用 at -m 这个参数来强制 at 传送一个执行完毕的 email 讯息给你自己喔！ ^_^

另外一个 at 的执行优点是什么呢？那就是『背景执行』的功能了！什么是背景执行啊？！ 很难了解吗？没关系，鸟哥提我自己的几个例子来给您听听，您就瞭了！

- 由于很多时候，我们其实都是使用 network 连接到主机来进行工作的， 但是 Client 与 Server 之间的网络联机其实并不见得很稳定，尤其是当你的 Client 计算机很忙的时候。 此时，万一我要进行一项长时间的工作时，那么风险就很大！鸟哥当初刚刚玩 Unix 时， 由于鸟哥所在的办公室太小了，无法有多个屏幕与键盘，因此，我都是利用我的 windows 98 再以网络联机软件连到 Unix 主机内作业的。当时我跑一个程序要跑 3 天...... 而众所皆知的， Windows 98 的长时间开机的稳定性确实..... 在某一次执行时，发生了.....剩下 3 个钟头就跑完却『联机终止』的情况～呜呜呜呜～ 又得要跑三天....
- 另一个常用的时刻则是例如上面的范例二，由于某个突发状况导致你必须要进行某项工作时， 这个 at 就很好用啦！

由于 at 工作排程的使用上，系统会将该项 at 工作独立出你的 bash 环境中， 直接交给系统的 atd 程序来接管，因此，当你下达了 at 的工作之后，就可以立刻离线了， 剩下的工作就完全交给 Linux 管理即可！所以啰，如果有长时间的网络工作时，嘿嘿！ 使用 at 可以让你免除网络断线后的困扰喔！ ^_^

那么万一我下达了 at 之后，才发现指令输入错误，该如何是好？呵呵！就将他移除啊！ 利用 atq 与 atrm 吧！

```
[root@linux ~]# atq
[root@linux ~]# atrm [jobnumber]

范例一：查询目前主机上面有多少的 at 工作排程？
[root@linux ~]# atq
10      2005-09-15 23:00 a root
# 上面说的是：『在 2005/09/15 的 23:00 有一项工作，该项工作指令下达者为
# root』而且，该项工作的工作号码 (jobnumber) 为 10 号喔！

范例二：将上述的第 10 个工作移除！
[root@linux ~]# atrm 10
[root@linux ~]# atq
# 没有任何信息，表示该工作被移除了！
```

利用 atq 与 atrm 来控制这个 at 的工作吧！ ^_^

循环执行的例行性命令

相对于 at 是仅执行一次的工作，循环执行的例行性命令则是由 cron (crond) 这个系统服务来控制的。由于系统预设就有相当多的例行性工作，因此，这个系统服务是预设启动的。另外， 由于使用者自己也可以进行例行性工作排程，所以啰， Linux 也提供使用者控制例行性命令的指令 (crontab)。 底下我们分别来聊一聊啰！

使用者的设定： crontab

使用者想要建立例行性命令时，使用的是 crontab 这个指令啦～不过，为了安全性的问题， 与 at 同样的，我们可以限制使用 crontab 的使用者账号喔！使用的限制数据有：

- /etc/cron.allow：
  将可以使用 crontab 的账号写入其中，若不在这个档案内的使用者则不可使用 crontab；
- /etc/cron.deny：
  将不可以使用 crontab 的账号写入其中，若未记录到这个档案当中的使用者，就可以使用 crontab 。

与 at 很像吧！同样的，以优先级来说， /etc/cron.allow 比 /etc/cron.deny 要优先， 而判断上面，这两个档案只选择一个来限制而已，因此，建议您只要保留一个即可， 免得影响自己在设定上面的判断！一般来说，系统预设是保留 /etc/cron.deny ， 您可以将不想让他执行 crontab 的那个使用者写入 /etc/cron.deny 当中，一个账号一行！

当使用者使用 crontab 这个指令来建立工作排程之后，该项工作就会被纪录到 /var/spool/cron/ 里面去了，而且是以账号来作为判别的喔！举例来说， dmtsai 使用 crontab 后， 他的工作会被纪录到 /var/spool/cron/dmtsai 里头去！但请注意，不要使用 vi 直接编辑该档案， 因为可能由于输入语法错误，会导致无法执行 cron 喔！另外， cron 执行的每一项工作都会被纪录到 /var/log/cron 这个登录档中，所以啰，如果您的 Linux 不知道有否被植入木马时， 也可以搜寻一下 /var/log/cron 这个登录档呢！

好了，那么我们就来聊一聊 crontab 的语法吧！

```
[root@linux ~]# crontab [-u username] [-l|-e|-r]
参数：
-u  ：只有 root 才能进行这个任务，亦即帮其它使用者建立/移除 crontab；
-e  ：编辑 crontab 的工作内容
-l  ：查阅 crontab 的工作内容
-r  ：移除 crontab 的工作内容
范例：

范例一：用 dmtsai 在每天的 12:00 发信给自己
[dmtsai@linux ~]$ crontab -e
# 此时会进入 vi 的编辑画面让您编辑工作！注意到，每项工作都是一行。
0   12  *  *  * mail dmtsai -s "at 12:00" < /home/dmtsai/.bashrc
#分 时 日 月 周 |<==============指令串========================>|
```

任何使用者只要不被列入 /etc/cron.deny 当中，那么他就可以直接下达『 crontab -e 』去编辑自己的例行性命令了！整个过程就如同上面提到的，会进入 vi 的编辑画面， 然后以一个工作一行来编辑，编辑完毕之后，输入『 :wq 』储存后离开 vi 就可以了！ 而每项工作的格式都是『 五个时间参数 实际动作指令 』，那么那五个时间参数代表什么呢？

代表意义	分钟	小时	日期	月份	周
数字范围	0-59	0-23	1-31	1-12	0-7

比较有趣的是那个『周』喔！当周为 0 或 7 时，都代表『星期天』的意思！另外， 还有一些辅助的字符，大概有底下这些：

特殊字符	代表意义
*	代表任何时刻都接受的意思！举例来说，上表的范例一，那个日、月、周都是 * ， 就代表着『不论何月、何日的礼拜几的 12:00 都执行后续指令』的意思！
,	代表分隔时段的意思。举例来说，如果要下达的工作是 3:00 与 6:00 时，就会是： 0 3,6 * * * command 还是有五栏，不过第二栏是 3,6 ，代表 3 与 6 都适用！
-	代表一段时间范围内，举例来说， 8 点到 12 点之间的每小时的 20 分都进行一项工作： 20 8-12 * * * command

	仔细看到第二栏变成 8-12 喔！代表 8,9,10,11,12 都适用的意思！
/n	那个 n 代表数字，亦即是『每隔 n 单位间隔』的意思，例如每五分钟进行一次，则： */5 * * * * command 很简单吧！用 * 与 /5 来搭配，也可以写成 0-59/5 ，相同意思！

我们就来搭配几个例子练习看看吧！

例题：假若你的女朋友生日是 5 月 2 日，你想要在 5 月 1 日的 23:59 发一封信给他，这封信的内容已经写在 /home/dmtsai/lover.txt 内了，该如何进行？
答：

直接下达 crontab -e 之后，编辑成为：

```
59 23 1 5 * mail kiki < /home/dmtsai/lover.txt
```

那样的话，每年 kiki 都会收到你的这封信喔！（当然啰，信的内容就要每年变一变啦！）

例题：假如每五分钟需要执行 /home/dmtsai/test.sh 一次，又该如何？
答：

同样使用 crontab -e 进入编辑：

```
*/5 * * * * /home/dmtsai/test.sh
```

那个 crontab 每个人都只有一个档案存在，就是在 /var/spool/cron 里面啊！ 还有建议您：『指令下达时，最好使用绝对路径，这样比较不会找不到执行档喔！』

例题：假如你每星期六都与朋友有约，那么想要每个星期五下午 4:30 告诉你朋友星期六的约会不要忘记，则：
答：

还是使用 crontab -e 啊！

```
30 16 * * 5 mail friend@his.server.name < /home/dmtsai/friend.txt
```

真的是很简单吧！呵呵！那么，该如何查询使用者目前的 crontab 内容呢？ 我们可以这样来看看：

```
[dmtsai@linux ~]$ crontab -l
59 23 1 5 * mail kiki < /home/dmtsai/lover.txt
*/5 * * * * /home/dmtsai/test.sh
30 16 * * 5 mail friend@his.server.name < /home/dmtsai/friend.txt

# 注意，若仅想要移除一项工作而已的话，必须要用 crontab -e 去编辑～
# 如果想要全部的工作都移除，才使用 crontab -r 喔！
[dmtsai@linux ~]$ crontab -r
[dmtsai@linux ~]$ crontab -l
no crontab for dmtsai
```

看到了吗？ crontab 『整个内容都不见了！』所以请注意：『如果只是要删除某个 crontab 的工作项目，那么请使用 crontab -e 来重新编辑即可！』如果使用 -r 的参数，是会将所有的 crontab 数据内容都删掉的！千万注意了！

系统的设定： /etc/crontab

这个『 crontab -e 』是针对使用者的 cron 来设计的，如果是『系统的例行性任务』时， 该怎么办呢？是否还是需要以 crontab -e 来管理你的例行性命令呢？当然不需要，你只要编辑 /etc/crontab 这个档案就可以啦！有一点需要特别注意喔！那就是 crontab -e 这个 crontab 其实是 /usr/bin/crontab 这个执行档，但是 /etc/crontab 可是一个『纯文字文件』喔！你可以 root 的身份编辑一下这个档案哩！

基本上， cron 这个服务的最低侦测限制是『分钟』，所以『 cron 会每分钟去读取一次 /etc/crontab 与 /var/spool/cron 里面的数据内容 』，因此，只要你编辑完 /etc/crontab 这个档案，并且将他储存之后，呵呵！那么 cron 的设定就自动的会来执行了！

Tips:
在 Linux 底下的 crontab 会自动的帮我们每分钟重新读取一次 /etc/crontab 的例行工作事项，但是某些原因或者是其它的 Unix 系统中，由于 crontab 是读到内存当中的，所以在你修改完 /etc/crontab 之后，可能并不会马上执行， 这个时候请重新启动 crond 这个服务吧！

    /etc/init.d/crond restart



好了，我们就来看一下这个 /etc/crontab 的内容吧！

```
[root@linux ~]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01  *  *  *  *   root      run-parts /etc/cron.hourly    <==每小时
02  4  *  *  *   root      run-parts /etc/cron.daily     <==每天
22  4  *  *  0   root      run-parts /etc/cron.weekly    <==每周日
42  4  1  *  *   root      run-parts /etc/cron.monthly   <==每个月 1 号
分 时 日 月 周 执行者身份  指令串
```

看到这个档案的内容你大概就了解了吧！呵呵，没错！这个档案与将刚刚我们下达 crontab -e 的内容几乎完全一模一样！只是有几个地方不太相同：

- MAILTO=root：
  这个项目是说，当 /etc/crontab 这个档案中的例行性命令发生错误时，或者是该执行结果有 STDOUT/STDERR 时， 会将错误讯息或者是屏幕显示的讯息传给谁？预设当然是由系统直接寄发一封 mail 给 root 啦！不过， 由于 root 并无法在客户端中以 POP3 之类的软件收信，因此，我通常都将这个 e-mail 改成自己的账号，好让我随时了解系统的状况！例如：
  MAILTO=dmtsai@my.host.name

- PATH=....：
  还记得我们在 BASH Shell 当中一直提到的执行文件路径问题吧！没错啦！这里就是输入执行文件的搜寻路径！ 使用预设的路径设定就已经很足够了！
- 01 * * * * root run-parts /etc/cron.hourly：
  在批注符号『 #run-parts 』这一行以后的命令，我们可以发现，五个数字后面接的是 root 喔！没错，与 crontab -e 的内容是不太一样的！这个字段的 root 代表的是『 执行的使用者身份为 root 』当然啰，你 也可以将这一行改写成其它的身份哩！而 run-parts 代表后面接的 /etc/cron.hourly 是『 一个目录内（/etc/cron.hourly）的所有可执行档 』，这也就是说，每个小时的 01 分，系统会以 root 的身份去 /etc/cron.hourly/ 这个目录下执行所有可以执行的档案！后面的三行也都是类似的意思！你可以到 /etc/ 底下去看看，系统本来就预设了这四个目录了！你可以将每天需要执行的命令直接写到 /etc/cron.daily/ 即可，还不需要使用到 crontab -e 的程序呢！方便吧！

基本上， /etc/crontab 这个档案里面支持两种下达指令的方式，一种是直接下达指令， 一种则是以目录来规划，例如：

- 指令型态
  01 * * * * dmtsai mail -s "testing" kiki < /home/dmtsai/test.txt
  以 dmtsai 这个使用者的身份，在每小时执行一次 mail 指令。
- 目录规划
  */5 * * * * root run-parts /root/runcron
  建立一个 /root/runcron 的目录，将要每隔五分钟执行的『可执行档』都写到该目录下， 就可以让系统每五分钟执行一次该目录下的所有可执行档。

这样就可以晓得 run-parts 的用意了吧！此外，与 crontab -e 规划当中最不相同的就是多了一个『使用者层级』的概念，通常我们都是以 root 的角度来规划例行性命令，但是总有不需要 root 的指令吧！就可以使用这个层级来规范该程序的使用者属于谁啰！

好！你现在大概了解了这一个咚咚吧！OK！假设你现在要作一个目录，让系统可以每 2 分钟去执行这个目录下的所有可以执行的档案，你可以写下如下的这一行在 /etc/crontab 中：

    */2 * * * * root run-parts /etc/cron.min

当然啰， /etc/cron.min 这个目录是需要存在的喔！那如果我需要执行的是一个『程序』而已， 不需要用到一个目录呢？该如何是好？例如在侦测网络流量时，我们希望每五分钟侦测分析一次， 可以这样写：

    */5 * * * * root /bin/mrtg /etc/mrtg/mrtg.cfg

没有了 run-parts 就是代表『一个档案』的意思啦！

如何！？建立例行性命令很简单吧！如果你是系统管理员的话，直接修改 /etc/crontab 这个档案即可喔！又便利，又方便管理呢！

一些使用特点：

有的时候，我们以系统的 cron 来进行例行性工作的建立时，要注意一些使用方面的特性。 举例来说，如果我们有四个工作都是五分钟要进行一次的，那么是否这四个动作全部都在同一个时间点进行？ 如果同时进行，该四个动作又很耗系统资源，如此一来，每五分钟不是会让系统忙得要死？ 呵呵！此时好好的分配一些执行时间，呵呵！就 OK 啦！所以，注意一下：

- 资源分配不均的问题：

当大量使用 crontab 的时候，总是会有问题发生的，最严重的问题就是『系统资源分配不均』的问题， 以鸟哥的系统为例，我有侦测流量的信息，包括：

- 流量
- 区域内其它 PC 的流量侦测
- CPU 使用率
- RAM 使用率
- 在线人数实时侦测

如果每个流程都在同一个时间启动的话，呵呵！那么在某个时段时，我的系统会变的相当的繁忙， 所以，这个时候就必须要分别设定啦！我可以这样做：

```
[root@linux ~]# vi /etc/crontab
1,6,11,16,21,26,31,36,41,46,51,56 * * * * root  CMD1
2,7,12,17,22,27,32,37,42,47,52,57 * * * * root  CMD2
3,8,13,18,23,28,33,38,43,48,53,58 * * * * root  CMD3
4,9,14,19,24,29,34,39,44,49,54,59 * * * * root  CMD4
```

看到了没？那个『 , 』分隔的时候，请注意，不要有空格符！（连续的意思）如此一来， 则可以将每五分钟工作的流程分别在不同的时刻来工作！则可以让系统的执行较为顺畅呦！

- 取消不要的输出项目：

另外一个困扰发生在『 当有执行成果或者是执行的项目中有输出的数据时，该数据将会 mail 给 MAILTO 设定的账号 』，好啦，那么当有一个排程一直出错（例如 DNS 的侦测系统当中，若 DNS 上层主机挂掉，那么你就会一直收到错误讯息！）怎么办？呵呵！还记得 BASH 与 Shell scripts 那一章吧！？直接以『命令重导向』将输出的结果输出到 /dev/null 这个垃圾桶当中就好了！

- 安全的检验：

很多时候被植入木马都是以例行命令的方式植入的，所以可以藉由检查 /var/log/cron 的内容来视察是否有『非您设定的 cron 被执行了？』这个时候就需要小心一点啰！

本章习题练习

（ 要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看 ）

- 今天假设我有一个指令程序，名称为： ping.sh 这个档名！我想要让系统每三分钟执行这个档案一次， 但是偏偏这个档案会有很多的讯息显示出来，所以我的 root 账号每天都会收到差不多四百多封的信件，光是收信就差不多快要疯掉了！ 那么请问应该怎么设定比较好呢？

  这个涉及命令重导向的问题，我们可以将他导入档案或者直接丢弃！如果该讯息不重要的话， 那么就予以丢弃，如果讯息很重要的话，才将他保留下来！假设今天这个命令不重要， 所以将他丢弃掉！因此，可以这样写：

  ```
  */5 * * * * root /usr/local/ping.sh > /dev/null 2>&1
  ```

- 您预计要在 2006 年的 2 月 14 日寄出一封给 kiki ，只有该年才寄出！该如何下达指令？

at 1am 2006-02-14

- 下达 crontab -e 之后，如果输入这一行，代表什么意思？
  * 15 * * 1-5 /usr/local/bin/tea_time.sh

  在每星期的 1~5 ，下午 3 点的每分钟，共进行 60 次 /usr/local/bin/tea_time.sh 这个档案。要特别注意的是，每个星期 1~5 的 3 点都会进行 60 次ㄟ！很麻烦吧～是错误的写法啦～ 应该是要写成：
  30 15 * * 1-5 /usr/local/bin/tea_time.sh

- 我用 vi 编辑 /etc/crontab 这个档案，我编辑的那一行是这样的：
  25 00 * * 0 /usr/local/bin/backup.sh
  这一行代表的意义是什么？

  这一行代表......没有任何意义！因为语法错误！您必须要了解，在 /etc/crontab 当中每一行都必须要有使用者才行！所以，应该要将原本那行改成：
  25 00 * * 0 root /usr/local/bin/backup.sh

- 请问，您的系统每天、每周、每个月各有进行什么工作？

  因为 FC4 系统预设的例行性命令都放置在 /etc/cron.* 里面，所以，你可以自行去：/etc/cron.daily/, /etc/cron.week/, /etc/cron.monthly/ 这三个目录内看一看， 就知道啦！^_^

- 每个星期六凌晨三点去系统搜寻一下内有 SUID/SGID 的任何档案！并将结果输出到 /tmp/uidgid.files

  vi /etc/crontab
  0 3 * * 6 root find / -perm +6000 > /tmp/uidgid.files

程序与资源管理

最近更新日期：2005/09/18

在 Linux 当中，Linux 是如何分辨一个程序的呢？嗯！当我们的系统里面有太多的死亡的程序的时候，应该怎样将该程序查出来之后并杀掉他呢？如果主机仅允许一次登入一个终端机画面，如何从事多个工作的进行呢？还有，如何设定一个程序，让他的执行顺序可以比较快速呢？！ 这个都是过程控制的重点项目呦！呵呵！另外一个又更常发生啦！如果我的 X-Window 死掉了！但是我的 Linux 基本上却还是活着的时候，那么是否需要重新 reboot 呢？还是有其它的方式可以重新启动 X-Window ？ 仔细瞧一瞧整个 process 的概念喔！

什么是程序 (process)

由前面一连几个章节的数据看来，我们一直强调在 Linux 底下所有的指令与您能够进行的动作都与权限有关， 而系统如何判定你的权限呢？当然就是前面 账号管理 章节当中提到的 UID/GID 的相关概念，以及档案的属性相关性啰！再进一步来解释， 您现在大概知道，在 Linux 系统当中：『触发任何一个事件时，系统都会将他定义成为一个程序，并且给予这个程序一个 ID ，称为 PID，同时依据启发这个程序的使用者与相关属性关系， 给予这个 PID 一组有效的权限设定。』 从此以后，这个 PID 能够在系统上面进行的动作，就与这个 PID 的权限有关了！

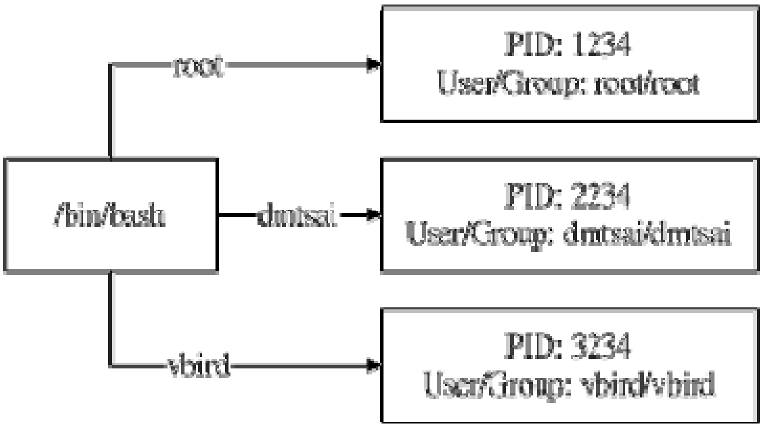
看这个定义似乎没有什么很奇怪的地方，不过，您得要了解什么叫做『触发事件』才行啊！ 我们在什么情况下会触发一个事件？而同一个事件可否被触发多次？呵呵！来了解了解先！

程序与执行文件 (process & program)

我们如何产生一个 Process ID (PID) 呢？其实很简单啦，就是『执行一个程序或指令』 就可以触发一个事件了而取得一个 PID 啰！我们说过，系统应该是仅认识 binary file 的， 那么当我们要让系统工作的时候，当然就是需要启动一个 binary file 啰， 那个 binary file 就是程序 (program) 啦！

那我们知道，每个程序都有三组人马的权限，每组人马都具有 r/w/x 的权限， 所以：『不同的使用者身份执行这个 program 时，系统给予的权限也都不相同！』 举例来说，我们可以利用 touch 来建立一个空的档案，当 root 执行这个 touch 指令时，他取得的是 UID/GID = 0/0 的权限，而当 dmtsai (UID/GID=501/501) 执行这个 touch 时，他的权限就跟 root 不同啦！

再举个更常见的例子，我们要操作系统的时候，通常是利用联机程序或者直接在主机前面登入， 然后取得我们的 shell 对吧！那么，我们的 shell 是 bash 对吧，这个 bash 在 /bin/bash 对吧， 那么同时间的每个人登入都是执行 /bin/bash 对吧！不过，每个人取得的权限就是不同！ 也就是说，我们可以这样看：



图一、程序与程序之间的差异

也就是说，当我们登入并执行 bash 时，系统已经给我们一个 PID 了， 这个 PID 就是依据登入者的 UID/GID (/etc/passwd) 来的啦～ 以上面的图来做说明的话，我们知道 /bin/bash 是一个程序 (program)，当 dmtsai 登入后，他取得一个 PID 号码为 2234 的程序，这个程序的 User/Group 都是 dmtsai， 而当这个程序进行其它作业时，例如上面提到的 touch 这个指令时， 那么由这个程序衍生出来的其它程序在一般状态下，也会沿用这个程序的相关权限的！

- 子程序与父程序：

在上面的说明里面，我们有提到所谓的『衍生出来的程序』，那是个啥咚咚？ 这样说好了，当我们登入系统后，会取得一个 bash 的 shell ，然后，我们用这个 bash 提供的接口去执行另一个指令，例如 /usr/bin/passwd 或者是 touch 等等， 那些另外执行的指令也会被触发成为 PID ，呵呵！那个 PID 就是『子程序』了， 而在我们的 bash 环境下，就称为『父程序』了！

另外，是否还记得我们在 bash shell 那一篇里面有提到 『环境变量』在不同程序之间的呼叫呢？现在稍微晓得是什么意思了吗？ 是啦！因为我们有执行不同的 bash 嘛！既然执行两次，自然就会取得两个 PID，而因为要让两个 PID 之间具有一些相关性，我们的 bash 就使用了环境变量啰！

例题：请在目前的 bash 环境下，再触发一次 bash ，并以『 ps -l 』这个指令观察程序相关的输出信息。
答：
　　直接执行 bash ，会进入到子程序的环境中，然后输入 ps -l 后，出现：

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   500 21337 21336  0  75   0 -  1348 wait   pts/1    00:00:00 bash
0 S   500 22573 21337  2  75   0 -  1348 wait   pts/1    00:00:00 bash
0 R   500 22591 22573  0  76   0 -  1302 -      pts/1    00:00:00 ps
有看到那个 PID 与 PPID 吗？第一个 bash 的 PID 与第二个 bash 的 PPID 都是
21337 啊，因为第二个 bash 是来自于第一个所产生的嘛！
```

重点来啦!所以说,在系统上面的各个程序可能是有相关性的喔！也就是有所谓的父程序与子程序的关系～至于程序的相关性，我们可以使用 pstree 这支程序去查验， 就能知道彼此之间的关系了。

另外要注意的是：所谓『擒贼先擒王』， 如果哪天你一直发现『奇怪，怎么有个程序关闭后，不久又会自动产生？ 而且自动产生的 PID 还不一样！』，呵呵！大概不需要怀疑的是，如果不是 例行性命令 的影响， 肯定有一支父程序存在，他会一直重新触发你想要关闭的那个程序， 导致你老是关不了。那怎么办？不是说过擒贼先擒王吗？关闭那支父程序啦！ ^_^

其实子程序与父程序之间的关系还挺复杂的，最大的复杂点在于程序互相之间的呼叫， 以及两者权限的相关性！这个可能就得要您自己多多建立自己的经验了～

- 系统或网络服务：常驻在内存的程序

如果就我们之前学到的一些指令数据来看，其实我们下达的指令都很简单， 包括用 ls 显示档案啊、用 touch 建立档案啊、rm/mkdir/cp/mv 等指令管理档案啊、 chmod/chown/passwd 等等的指令来管理权限等等的，不过，这些指令都是执行完就结束了。 也就是说，该项指令被触发后所产生的 PID 很快就会终止呢！ 那有没有一直在执行的程序啊？当然有啊！而且多的是呢！

举个简单的例子来说好了，我们知道系统每分钟都会去扫瞄 /etc/crontab 以及相关的设定档， 来进行工作排程吧？那么那个工作排程是谁负责的？当然不是鸟哥啊！ 呵呵！是 crond 这个程序所管理的，我们将他启动在背景当中一直持续不断的运作， 套句以前 DOS 年代常常说的一句话，那就是『常驻在内存当中的程序』啦！

这些常驻在内存当中的程序有很多，不过主要大致分成系统本身所需要的服务，例如刚刚提到的 crond 及 atd ，还有 syslog 等等的。还有一些则是负责网络联机的服务，例如 Apache, named, postfix, vsftpd... 等等的。这些网络服务比较有趣的地方， 在于这些程序被执行后，他会启动一个可以负责网络监听的端口口 (port) ， 以提供外部客户端 (client) 的联机要求。

这部分我们会在认识系统服务的地方再好好的讲一讲， 在这里，您先有个概念，大概知道一下，系统上面的 PID 几乎都是透过执行一些指令所产生的， 而这些指令可能会负责一些重要的工作，例如网络服务器啊、系统效能维持啊、 或是其它系统重要工作等等。若有兴趣的话，可以先以 netstat 检查一下您主机上的网络服务喔！

Linux 的多人多任务环境

我们现在知道了，其实在 Linux 底下执行一个指令时，系统会给予这个动作一个 ID， 我们称为 PID，而根据启用这个指令的使用者与相关的指令功能，而给予这个特定 PID 一组权限， 该指令可以进行的行为

则与这个 PID 的权限有关。根据这个说明，我们就可以简单的了解， 为什么 Linux 这么多用户，但是却每个人都可以拥有自己的环境了吧！^_^

- 多人环境：

Linux 最棒的地方就在于他的多人多任务环境了！那么，什么是『多人多任务』？！在 Linux 上面允许不同的人使用，而且每个人都有其特殊的权限，只有一个人具有至高无上的权力，那就是 root（系统管理员），除了他之外，其它人都必须要受一些限制的！而每个人进入 Linux 的环境设定都可以随着每个人的喜好来设定 (还记得我们在 BASH 那一章提过的 ~/.bashrc 吧！？对了！就是那个光！)！现在知道为什么了吧？因为每个人登入后取得的 shell 的 PID 不同嘛！

- 多任务行为：

我想，在某些其它操作系统中，您可能会遇到这样的事情：『这个档案正在使用中， 您无法开启这个档案！』我哩勒！还得将正在执行当中的程序关掉之后才能开这个中间暂存档！！ 而且这个时候还只有我自己一个人在使用呢～受不了～呵呵！不过， Linux 就不会这样啰！您可以同时在不同的画面，同时由不同的人 (当然啰，有的是经由 SSH 网络联机过来，有的是直接在屏幕前面的朋友啰！)使用『同一个档案』， 不论是开启或者是修改，只要您有权限，就可以使用该档案！！

这个东西可有用的紧！由于鸟哥是很常使用程序的 (就是 Fortran 啦，吃饭的工具！) ，而由于我们有一部主机专门用来工作的，所以配备比较高档一点 PIII 的双 CPU)，那么我就可以同时的进行两个 compiler 的程序，而且还不会互相的影响， 并且资源分配的还蛮均匀的！哈哈！怎么会跑得这么顺畅呀！爽毙了！

其实操作系统的多任务是很复杂的行为啦！尤其涉及将多个工作直接丢给一颗 CPU 来处理～ 现在我们应该比较清楚的是，所谓的『工作』其实是将多个指令触发成为系统程序 (PID)， 而这些程序若同时被触发时，那么一颗 CPU 就得要同时负责许多工作了。 但我们晓得的是，并非每个程序都很耗系统资源，例如前一节提到的 crond 与 atd 这两个系统服务， 他们并没有消耗很多系统资源的。此时，当然啰， CPU 就可以进行其它工作， 这就是所谓的多任务！

- 多重登入环境的七个基本终端窗口：

在 Linux 当中，预设提供了六个文字界面登入窗口，以及一个图形界面，你可以使用 [Alt]+[F1].....[F7] 来切换不同的终端机界面，而且每个终端机界面的登入者还可以不同人！ 很炫吧！这个东西可就很有用啦！尤其是在某个程序死掉的时候！

其实，这也是多任务环境下所产生的一个情况啦！ 我们的 Linux 预设会启动六个终端机登入环境的程序，所以我们就会有六个终端机接口。 您也可以减少啊！就是减少启动的终端机程序就好了。详细的资料可以先查阅 /etc/inittab 这个档案，未来我们在开机管理流程会再仔细的介绍的！

- 特殊的程序管理行为：

以前的鸟哥笨笨的，总是以为使用 Windows 98 就可以啦！后来，因为工作的关系，需要使用 Unix 系统，想说我只要在工作机前面就好， 才不要跑来跑去的到 Unix 工作站前面去呢！所以就使用 Windows 连到我的 Unix 工作站工作！

好死不死，我一个程序跑下来要 2~3 天，唉～偏偏常常到了第 2.5 天的时候， Windows 98 就给他挂点去！当初真的是给他怕死了～后来因为换了新计算机，用了随机版的 Windows 2000 ，呵呵，这东西真不错 (指对单人而言)，在当机的时候， 他可以仅将错误的程序踢掉，而不干扰其它的程序进行，呵呵！ 从

此以后，就不用担心会当机连连啰！不过，2000 毕竟还不够好，因为有的时候还是会死当！！

那么 Linux 呢？哈哈！更棒了，几乎可以说绝对不会当机的！因为他可以在任何时候， 将某个被困住的程序杀掉，然后在重新执行该程序而不用重新开机！够炫吧！那么如果我在 Linux 下以文字界面登入，在屏幕当中显示错误讯息后就挂了～动都不能动，该如何是好！？ 这个时候那预设的七个窗口就帮上忙啦！你可以随意的再按 [Alt]+[F1].....[F7] 来切换到其它的终端机界面，然后以 ps -aux 找出刚刚的错误程序，然后给他 kill 一下，哈哈，回到刚刚的终端机界面！恩～棒！又回复正常啰！

为什么可以这样做呢？我们刚刚不是提过吗？每个程序之间可能是独立的，也可能有相依性， 只要到独立的程序当中，删除有问题的那个程序，当然他就可以被系统移除掉啦！^_^

- bash 环境下的工作管理 (job control)

我们在上一个小节有提到所谓的『父程序、子程序』的关系，那我们登入 bash 之后， 就是取得一个名为 bash 的 PID 了，而在这个环境底下所执行的其它指令， 就几乎都是所谓的子程序了。那么，在这个单一的 bash 接口下，我可不可以进行多个工作啊？ 当然可以啦！可以『同时』进行喔！举例来说，我可以这样做：

```
[root@linux ~]# cp file1 file2 &
```

在这一串指令中，重点在那个 & 的功能，他表示将 file1 这个档案复制为 file2 ，且放置于背景中执行，也就是说执行这一个命令之后，在这一个终端接口仍然可以做其它的工作！而当这一个指令 (cp file1 file2)执行完毕之后，系统将会在您的终端接口显示完成的消息！很便利喔！

- 多人多任务的系统资源分配问题考虑：

多人多任务确实有很多的好处，但其实也有管理上的困扰，因为使用者越来越多， 将导致你管理上的困扰哩！另外，由于使用者日盛，当使用者达到一定的人数后， 通常你的机器便需要升级了，因为 CPU 的运算与 RAM 的大小可能就会不敷使用！

举个例子来说，鸟哥之前的网站管理的有点不太好，因为使用了一个很复杂的人数统计程序， 这个程序会一直去取用 MySQL 数据库的数据，偏偏因为流量大，造成 MySQL 很忙碌。 在这样的情况下，当鸟哥要登入去写网页数据，或者要去使用讨论区的资源时， 哇！慢的很！简直就是『龟速』啊！后来终于将这个程序停止不用了， 以自己写的一个小程序来取代，呵呵！这样才让 CPU 的负载 (loading) 整个降下来～ 用起来顺畅多了！ ^_^

好了！废话说完了！开始来谈一谈几个常用的指令吧！

工作管理 (job control)： &, [ctrl]-z, jobs, fg, bg, kill

这个工作管理 (job control) 是用在 bash 环境下的，也就是说：『 当我们登入系统取得 bash shell 之后，在单一终端机接口下同时进行多个工作的行为管理 』。举例来说，我们在登入 bash 后， 想要一边复制档案、一边进行资料搜寻、一边进行编译，还可以一边进行 vi 程序撰写！ 当然我们可以重复登入那六个文字接口的终端机环境中，不过，能不能在一个 bash 内达成？ 当然可以啊！就是使用 job control 啦！ ^_^

从上面的说明当中，您应该要了解的是：『进行工作管理的行为中， 其实每个工作都是目前 bash 的子程序，亦即彼此之间是有相关性的。 我们无法以 job control 的方式由 tty1 的环境去管理 tty2 的

bash ！』 这个概念请您先建立起来，后续的范例介绍之后，您就会清楚的了解啰！

或许你会觉得很奇怪啊，既然我可以在六个终端接口登入，那何必使用 job control 呢？ 真是脱裤子放屁，多此一举啊！不要忘记了呢，我们可以在 /etc/security/limits.conf 里面设定使用者同时可以登入的联机数，在这样的情况下，某些使用者可能仅能以一个联机来工作呢！ 所以啰，您就得要了解一下这种工作管理的模式了！ 此外，这个章节内容也会牵涉到很多的数据流重导向，所以，如果忘记的话， 务必回到 BASH Shell 看一看喔！

总之，要进行 bash 的 job control 必须要注意到的限制是：

- 程序必须来是 shell 的 child process
- 程序不能等待 terminal/shell 的输入(input)

- 直接将指令丢到背景中『执行』的 & ：

瞎密？将某个指令『丢到背景』当中？在一个 bash 的环境下，什么叫做『前景 (foreground) 』与『背景 (background) 』？我们先来简单的定义一下：

- 前景：您可以控制的这个工作称为前景的工作 (foreground)；
- 背景：在内存内可以自行运作的工作，您无法直接控制他，除非以 bg/fg 等指令将该工作呼叫出来。

如同前面提到的，我们在只有一个 bash 的环境下，如果想要同时进行多个工作， 那么可以将某些工作丢到背景环境当中，让我们可以继续操作前景的工作！那么如何将工作丢到背景中？ 最简单的方法就是利用『 & 』这个玩意儿了！举个简单的例子， 我们要将 /etc/ 整个备份成为 /tmp/etc.tar.gz 时，又不想要等待， 那么可以这样做：

```
[root@linux ~]# tar -zpcf /tmp/etc.tar.gz /etc &
[1] 24874   <== [job number] PID
[root@linux ~]#    <== 可以继续作业，不受影响！这就是前景！
```

仔细的瞧一瞧，我在输入一个指令后，在该指令的最后面加上一个『 & 』代表将该指令丢到背景中， 此时 bash 会给予这个指令一个『工作号码(job number)』，就是那个 [1] 啦！ 至于后面那个 24874 则是该指令所触发的『 PID 』了！ 而且，有趣的是，我们可以继续操作 bash 呢！很不赖吧！ 不过，那么丢到背景中的工作什么时候完成？完成的时候会显示什么？ 如果你输入几个指令后，突然出现这个数据：

```
[1]+  Done                    tar -zpcf /tmp/etc.tar.gz /etc
```

就代表 [1] 这个工作已经完成 (Done) ，该工作的指令则是接在后面那一串指令列。 这样了解了吧？！ 另外，这个 & 代表：『将工作丢到背景中去执行』喔！ 注意到那个『执行』的字眼！此外，这样的情况最大的好处是： 不怕被 [ctrl]-c 中断的啦！

此外，将工作丢到背景当中要特别注意资料的流向喔！举例来说，如果我将刚刚那个指令改成：

```
[root@linux ~]# tar -zpcvf /tmp/etc.tar.gz /etc &
```

情况会怎样？呵呵，在背景当中执行的指令，如果有 stdout 及 stderr 时， 他的数据依旧是输出到屏幕上面的，所以，我们会无法看到提示字符，当然也就无法完好的掌握前景工作。 所以啰，最佳的状况就是利用数据流重导向，将输出数据传送至某个档案中。举例来说，我可以这样做：

```
[root@linux ~]# tar -zpcvf /tmp/etc.tar.gz /etc > /tmp/log.txt 2>&1 &
[1] 24984
[root@linux ~]#
```

呵呵！如此一来，数据都给他传送到 /tmp/log.txt 当中，当然就不会影响到我们前景的作业了。 这样说，您应该可以更清楚数据流重导向的重要性了吧？！ ^_^

- 将『目前』的工作丢到背景中『暂停』：[ctrl]-z

想个情况：如果我正在使用 vi ，却发现我有个档案不知道放在哪里，需要到 bash 环境下去搜寻，此时，是否要结束 vi 呢？呵呵！当然不需要啊！只要暂时将 vi 给他丢到背景当中等待即可。 例如以下的案例：

```
[root@linux ~]# vi ~/.bashrc
# 在 vi 的一般模式下，按下 [ctrl]-z 这两个按键
[1]+  Stopped                 /usr/bin/vim ~/.bashrc
[root@linux ~]#   <==顺利取得了前景的操控权！
```

在 vi 的一般模式下，按下 [ctrl] 及 z 这两个按键，屏幕上会出现 [1] ，表示这是第一个工作， 而那个 + 代表目前在背景下预设被取用的那个工作 (与 fg 这个指令有关)！ 而那个 Stopped 则代表目前这个工作的状态。在预设的情况下，使用 [ctrl]-z 丢到背景当中的工作都是『暂停』的状态喔！

- 观察目前的背景工作状态： jobs

```
[root@linux ~]# jobs [-lrs]
参数：
-l  ：除了列出 job number 之外，同时列出 PID
-r  ：仅列出正在背景 run 的工作；
-s  ：仅列出正在背景当中暂停 (stop) 的工作。
范例：

范例一：观察目前的 bash 当中，所有的工作，与对应的 PID
[root@linux ~]# jobs -l
[1]+ 24988 Stopped                 /usr/bin/vim ~/.bashrc
[2]- 25006 Stopped                 /usr/bin/vim ~/.bash_history
```

如果想要知道目前有多少的工作在背景当中，就用 jobs 这个指令吧！一般来说，直接下达 jobs 即可！ 不过，如果您还想要知道该 job number 的 PID 号码，可以加上 -l 这个参数啦！ 在输出的信息当中，例如上表，仔细看到那个 +- 号喔！ 那个 + 代表预设的取用工作。 所以说：『目前我有两个工作在背景当中，两个工作都是暂停的， 而如果我仅输入 fg 时，那么那个 [1] 会被拿到前景当中来处理』！

- 将背景工作拿到前景来处理：fg

刚刚提到的都是将工作丢到背景当中去执行的，那么有没有可以将背景工作拿到前景来处理的？ 有啊！就是那个 fg 啦！举例来说，我们想要将上头范例当中的工作拿出来处理时：

```
[root@linux ~]# fg %jobnumber
参数：
%jobnumber ：工作的号码。注意，那个 % 是可有可无的！
范例：

范例一：先以 jobs 观察工作，再将工作取出：
[root@linux ~]# jobs
[1]+  Stopped                 /usr/bin/vim ~/.bashrc
[2]-  Stopped                 /usr/bin/vim ~/.bash_history
[root@linux ~]# fg      <==预设取出那个 + 的工作，亦即 [1]
[root@linux ~]# fg %2   <==直接规定取出的那个工作号码！
```

经过 fg 指令就能够将背景工作拿到前景来处理啰！

- 让工作在背景下进行： bg

我们刚刚提到，那个 [ctrl]-z 可以将目前的工作丢到背景底下去『暂停』， 那么如何让一个工作在背景底下『 Run 』呢？我们可以在底下这个案例当中来测试！ 注意喔！底下的测试要进行的快一点！^_^

```
范例一：一执行 find / -perm +7000 后，立刻丢到背景去暂停！
[root@linux ~]# find / -perm +7000
# 此时，请立刻按下 [ctrl]-z 暂停！
[1]+  Stopped                 find / -perm +7000
[root@linux ~]#

范例二：让该工作在背景下进行，并且观察他！！
[root@linux ~]# jobs ; bg %1 ; jobs
[1]+  Stopped                 find / -perm +7000
[1]+ find / -perm +7000 &
[1]+  Running                 find / -perm +7000 &
```

看到哪里有差异吗？呼呼！没错！就是那个状态列～以经由 Stopping 变成了 Running 啰！ 看到差异点，嘿嘿！指令列最后方多了一个 & 的符号啰！ 代表该工作被启动在背景当中了啦！ ^_^

- 管理背景当中的工作： kill

刚刚我们可以让一个已经在背景当中的工作继续工作，也可以让该工作以 fg 拿到前景来， 那么，如果想要将该工作直接移除呢？或者是将该工作重新启动呢？呵呵！ 这个时候就得需要给予该工作一个讯号 (signal) ，让他知道该怎么作才好啊！ 此时， kill 这个指令就派上用场啦！

```
[root@linux ~]# kill -signal %jobnumber
[root@linux ~]# kill -l
参数：
-l  ：这个是 L 的小写，列出目前 kill 能够使用的讯号 (signal) 有哪些？
```

```
signal ：代表给予后面接的那个工作什么样的指示啰！用 man 7 signal 可知：
  -1 ：重新读取一次参数的设定档 (类似 reload)；
  -2 ：代表与由键盘输入 [ctrl]-c 同样的动作；
  -9 ：立刻强制删除一个工作；
  -15：以正常的程序方式终止一项工作。与 -9 是不一样的。
范例：

范例一：找出目前的 bash 环境下的背景工作，并将该工作删除。
[root@linux ~]# jobs
[1]+  Stopped                 vim bashrc
[root@linux ~]# kill -9 %1
[1]+  已砍掉                  vim bashrc

范例：找出目前的 bash 环境下的背景工作，并将该工作终止掉。
[root@linux ~]# jobs
[1]+  Stopped                 vim bashrc
[root@linux ~]# kill -SIGTERM %1
[1]+  终止                    vim bashrc
# -SIGTERM 与 -15 是一样的！您可以使用 kill -l 来查阅！
```

特别留意一下， -9 这个 signal 通常是用在『强制删除一个不正常的工作』时所使用的， -15 则是以正常步骤结束一项工作(15 也是默认值)，两者之间并不相同吶！举上面的例子来说， 我用 vi 的时候，不是会产生一个 .filename.swp 的档案吗？ 那么，当使用 -15 这个 signal 时， vi 会尝试以正常的步骤来结束掉该 vi 的工作， 所以 .filename.swp 会主动的被移除，但若是使用 -9 这个 signal 时， 由于该 vi 工作会被强制移除掉，因此， .filename.swp 就会继续存在档案系统当中。 这样您应该可以稍微分辨一下了吧？

其实， kill 的妙用是很无穷的啦！他搭配 signal 所详列的信息 (用 man 7 signal 去查阅相关资料) 可以让您有效的管理工作与程序 (Process)，此外，那个 killall 也是同样的用法！ 至于常用的 signal 您至少需要了解 1, 9, 15 这三个 signal 的意义才好。 此外， signal 除了以数值来表示之外，也可以使用讯号名称喔！ 举例来说，上面的范例二就是一个例子啦！至于 signal number 与名称的对应， 呵呵，使用 kill -l 就知道啦(L 的小写)！

程序管理

本章一开始就提到所谓的『程序』的概念，包括程序的触发、子程序与父程序的相关性等等， 此外，还有那个『程序的相依性』以及所谓的『殭尸程序』等等需要说明的呢！ 为什么程序管理这么重要呢？

- 首先，由于我们在操作系统时，各项工作其实都是经过某个 PID 来达成的， 因此，能不能进行某项工作，就与该程序的权限有关了。
- 再来，如果您的 Linux 系统是个很忙碌的系统，那么当整个系统资源快要被使用光时， 您是否能够找出最耗系统的那个程序，然后删除该程序，让系统恢复正常呢？
- 此外，如果由于某个程序写的不好，导致产生一个有问题的程序在内存当中， 您又该如何找出他，然后将他移除呢？

* 如果同时有五六项工作在您的系统当中运作，但其中有一项工作才是最重要的， 该如何让那一项重要的工作被最优先执行呢？

所以啰，一个称职的系统管理员，必须要熟悉程序的管理流程才行，否则当系统发生问题时， 还真是很难解决问题呢！当然啦，程序的管理其实也是很难理解的部分， 尤其讲到子程序与父程序之间的关系，更不容易理解。伤脑筋啊！无论如何，咱们还是得来看一看， 系统上面到底有多少程序在运作啊？

程序的观察

既然程序这么重要，那么我们如何查阅系统上面正在运作当中的程序呢？很简单啊！ 利用静态的 ps 或者是动态的 top，还能以 pstree 来查阅程序树之间的关系喔！

---

* ps

```
[root@linux ~]# ps aux
[root@linux ~]# ps -lA
[root@linux ~]# ps axjf
参数：
-A  ：所有的 process 均显示出来，与 -e 具有同样的效用；
-a  ：不与 terminal 有关的所有 process ；
-u  ：有效使用者 (effective user) 相关的 process ；
x   ：通常与 a 这个参数一起使用，可列出较完整信息。
输出格式规划：
l   ：较长、较详细的将该 PID 的的信息列出；
j   ：工作的格式 (jobs format)
-f  ：做一个更为完整的输出。
特别说明：
由于 ps 能够支持的 OS 类型相当的多，所以他的参数多的离谱！
而且有没有加上 - 差很多！详细的用法应该要参考 man ps 喔！
范例：

范例一：将目前属于您自己这次登入的 PID 与相关信息列示出来
[root@linux ~]# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0  5881  5654  0  76   0 -  1303 wait   pts/0    00:00:00 su
4 S     0  5882  5881  0  75   0 -  1349 wait   pts/0    00:00:00 bash
4 R     0  6037  5882  0  76   0 -  1111 -      pts/0    00:00:00 ps
# 上面这个信息其实很多喔！各相关信息的意义为：
# F       代表这个程序的旗标 (flag)， 4 代表使用者为 super user；
# S       代表这个程序的状态 (STAT)，关于各 STAT 的意义将在内文介绍；
# PID     没问题吧！？就是这个程序的 ID 啊！底下的 PPID 则上父程序的 ID；
# C       CPU 使用的资源百分比
# PRI     这个是 Priority (优先执行序) 的缩写，详细后面介绍；
```

```
# NI      这个是 Nice 值，在下一小节我们会持续介绍。
# ADDR    这个是 kernel function，指出该程序在内存的那个部分。如果是个 running
#         的程序，一般就是『 - 』的啦！
# SZ      使用掉的内存大小；
# WCHAN   目前这个程序是否正在运作当中，若为 - 表示正在运作；
# TTY     登入者的终端机位置啰；
# TIME    使用掉的 CPU 时间。
# CMD     所下达的指令为何！？
# 仔细看到每一个程序的 PID 与 PPID 的相关性为何喔！上头列出的三个程序中，
# 彼此间可是有相关性的呐！

范例二：列出目前所有的正在内存当中的程序：
[root@linux ~]# ps aux
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  1740  540 ?        S    Jul25   0:01 init [3]
root         2  0.0  0.0     0    0 ?        SN   Jul25   0:00 [ksoftirqd/0]
root         3  0.0  0.0     0    0 ?        S<   Jul25   0:00 [events/0]
.....中间省略.....
root      5881  0.0  0.3  5212 1204 pts/0    S    10:22   0:00 su
root      5882  0.0  0.3  5396 1524 pts/0    S    10:22   0:00 bash
root      6142  0.0  0.2  4488  916 pts/0    R+   11:45   0:00 ps aux

范例三：以范例一的显示内容，显示出所有的程序：
[root@linux ~]# ps -lA
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  76   0 -   435 -      ?        00:00:01 init
1 S     0     2     1  0  94  19 -     0 ksofti ?        00:00:00 ksoftirqd/0
1 S     0     3     1  0  70  -5 -     0 worker ?        00:00:00 events/0
.....以下省略.....

范例四：列出类似程序树的程序显示：
[root@linux ~]# ps -axjf
PPID  PID PGID  SID TTY    TPGID STAT UID  TIME COMMAND
   0    1    0    0 ?         -1 S      0  0:01 init [3]
   1    2    0    0 ?         -1 SN     0  0:00 [ksoftirqd/0]
.....中间省略.....
   1 5281 5281 5281 ?         -1 Ss     0  0:00 /usr/sbin/sshd
5281 5651 5651 5651 ?         -1 Ss     0  0:00  \_ sshd: dmtsai [priv]
5651 5653 5651 5651 ?         -1 S    500  0:00      \_ sshd: dmtsai@pts/0
5653 5654 5654 5654 pts/0   6151 Ss   500  0:00          \_ -bash
5654 5881 5881 5654 pts/0   6151 S      0  0:00              \_ su
5881 5882 5882 5654 pts/0   6151 S      0  0:00                  \_ bash
5882 6151 6151 5654 pts/0   6151 R+     0  0:00                      \_ ps -axjf
```

```
# 看出来了吧？其实鸟哥在进行一些测试时，都是以网络联机进主机来测试的，
# 所以啰，你会发现，嘿嘿！其实程序之间是有相关性的啦！不过，
# 其实还可以使用 pstree 来达成这个程序树喔！底下在仔细谈一谈。

范例五：找出与 cron 与 syslog 这两个服务有关的 PID 号码？
[root@linux ~]# ps aux | egrep '(cron|syslog)'
root   1539  0.0  0.1  1616   616 ?        Ss   Jul25   0:03 syslogd -m 0
root   1676  0.0  0.2  4544  1128 ?        Ss   Jul25   0:00 crond
root   6157  0.0  0.1  3764   664 pts/0    R+   12:10   0:00 egrep (cron|syslog)
# 所以号码是 1539 及 1676 这两个啰！就是这样找的啦！
```

说真的，如果你曾经使用 man ps 的话，呵呵！可能会被里面的说明搞的一脸茫然～ 因为......支持的类型实在太多，所以， ps -aux 与 ps aux 显示的结果『可能』是不一样的， 那个 -u 后面接的是『有效的使用者 ID』，所以， -ux 可能是『有一个 user 名称为 x 』 而如果没有 x 这个使用者，那么屏幕上面会显示一个警告讯息，并以 ps aux 来输出。 哇！真是麻烦～所以，您可以直接记得使用 ps aux 就好了！

在预设的情况下， ps 仅会列出与目前所在的 bash shell 有关的 PID 而已，所以， 当我使用 ps -l 的时候，只有三个 PID (范例一)。注意一下，我有一个 bash 的 PID ， 而且也有一个 ps 的 PID ，了解了吧？呵呵！在 bash 里面执行程序时，是会触发一个新的 process 的喔！而且，两者之间是有相关性的，看 PID 与 PPID 的号码，你就会晓得两者的差异了！

那么，什么是『有效使用者 ID 』呢？还记得我们提过的 SUID 吧？我以 dmtsai 去执行 /usr/bin/passwd 取得的那个 process 竟然是 root 的权限喔！ 此时，实际的使用者 (real user) 是 dmtsai ，但是有效的使用者 (effective user) 是 root 啦！ 这样说，可以理解吧！？

一般来说，鸟哥会建议您，直接使用『ps aux』这个指令参数即可， 显示的结果例如上面的范例二啰。在范例二的各个显示项目代表的意义为：

- USER：该 process 属于那个使用者账号的？
- PID ：该 process 的号码。
- %CPU：该 process 使用掉的 CPU 资源百分比；
- %MEM：该 process 所占用的物理内存百分比；
- VSZ ：该 process 使用掉的虚拟内存量 (Kbytes)
- RSS ：该 process 占用的固定的内存量 (Kbytes)
- TTY ：该 process 是在那个终端机上面运作，若与终端机无关，则显示 ?，另外， tty1-tty6 是本机上面的登入者程序，若为 pts/0 等等的，则表示为由网络连接进主机的程序。
- STAT：该程序目前的状态，主要的状态有：
    - R ：该程序目前正在运作，或者是可被运作；
    - S ：该程序目前正在睡眠当中 (可说是 idle 状态啦！)，但可被某些讯号 (signal) 唤醒。
    - T ：该程序目前正在侦测或者是停止了；
    - Z ：该程序应该已经终止，但是其父程序却无法正常的终止他，造成 zombie (疆尸) 程序的状态
- START：该 process 被触发启动的时间；

- TIME ：该 process 实际使用 CPU 运作的时间。
- COMMAND：该程序的实际指令为何？

我们取这一行来做个简单的说明：

```
root      5881  0.0  0.3   5212  1204 pts/0    S    10:22   0:00 su
```

该程序属于 root 所有，他的 PID 号码是 5881，该程序对于 CPU 的使用率很低啊！ 至于占用的物理内存大概有 0.3% 这么多。至于该程序使用掉的虚拟内存量为 5212 K， 物理内存为 1204 K，该程序属于 pts/0 这个终端机，看来这个程序应该是来自网络的联机登入。 该程序目前是 Sleep 的状态，但其实是可以被执行的。这个程序由今天的 10:22 开始运作， 不过，仅耗去 CPU 运作时间的 0:00 分钟。该程序的执行就是 su 这个指令啦！

除此之外，我们必须要知道的是『疆尸 (zombie) 』程序是什么？ 通常，造成疆尸程序的成因是因为该程序应该已经执行完毕，或者是因故应该要终止了， 但是该程序的父程序却无法完整的将该程序结束掉，而造成那个程序一直存在内存当中..... 如果您发现在某个程序的 CMD 后面还接上 <defunct> 时，就代表该程序是疆尸程序啦，例如：

```
apache  8683  0.0  0.9 83384 9992 ?   Z  14:33   0:00 /usr/sbin/httpd <defunct>
```

当系统不稳定的时候就容易造成所谓的疆尸程序，可能原因是因为程序写的不好啦， 或者是使用者的操作习惯不良等等所造成。如果您发现系统中很多疆尸程序时， 呵呵！记得啊！要找出该程序的父程序，然后好好的做个追踪，好好的进行主机的环境最佳化啊！ 看看有什么地方需要改善的，不要只是直接将他 kill 掉而已呢！ 不然的话，万一他一直产生，那可就麻烦了！ @_@

- top

```
[root@linux ~]# top [-d] | top [-bnp]
参数：
-d  ：后面可以接秒数，就是整个程序画面更新的秒数。预设是 5 秒；
-b  ：以批次的方式执行 top ，还有更多的参数可以使用喔！
      通常会搭配数据流重导向来将批次的结果输出成为档案。
-n  ：与 -b 搭配，意义是，需要进行几次 top 的输出结果。
-p  ：指定某些个 PID 来进行观察监测而已。
在 top 执行过程当中可以使用的按键指令：
          ? ：显示在 top 当中可以输入的按键指令；
          P ：以 CPU 的使用资源排序显示；
          M ：以 Memory 的使用资源排序显示；
          N ：以 PID 来排序喔！
          T ：由该 Process 使用的 CPU 时间累积 (TIME+) 排序。
          k ：给予某个 PID 一个讯号  (signal)
          r ：给予某个 PID 重新制订一个 nice 值。
范例：

范例一：每两秒钟更新一次 top ，观察整体信息：
[root@linux ~]# top -d 2
```

```
top - 18:30:36 up 30 days, 7 min,  1 user,  load average: 0.42, 0.48, 0.45
Tasks: 163 total,   1 running, 161 sleeping,   1 stopped,   0 zombie
Cpu(s):  4.7% us,  4.0% sy,  6.3% ni, 82.5% id,  0.4% wa,  0.1% hi,  2.0% si
Mem:   1033592k total,   955252k used,    78340k free,   208648k buffers
Swap:  1052216k total,      728k used,  1051488k free,   360248k cached
    <==如果加入 k 或 r 时，就会有相关的字样出现在这里喔！

 3981 apache    34  19 84012  11m 7352 S 17.3  1.2   0:00.09 httpd
 1454 mysql     16   0  289m  40m 2228 S  3.8  4.0 115:01.32 mysqld
 3985 dmtsai    15   0  2148  904  668 R  3.8  0.1   0:00.03 top
    1 root      16   0  3552  552  472 S  0.0  0.1   0:08.90 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:52.76 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:03.01 ksoftirqd/0

范例二：将 top 的信息进行 2 次，然后将结果输出到 /tmp/top.txt
[root@linux ~]# top -b -n 2 > /tmp/top.txt
# 这样一来，嘿嘿！就可以将 top 的信息存到 /tmp/top.txt 档案中了。

范例三：假设 10604 是一个已经存在的 PID ，仅观察该程序？
[root@linux ~]# top -d 2 -p10604
top - 13:53:00 up 51 days,  2:27,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:   1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:    385676k total,   371760k used,    13916k free,   131164k buffers
Swap:  1020116k total,      880k used,  1019236k free,    95772k cached


10604 root      16   0  5396 1544 1244 S  0.0  0.4   0:00.07 bash

范例四：承上题，上面的 NI 值是 0 ，想要改成 10 的话？
# 在范例三的 top 画面当中直接按下 r 之后，会出现如下的图样！
top - 13:53:00 up 51 days,  2:27,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:   1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:    385676k total,   371760k used,    13916k free,   131164k buffers
Swap:  1020116k total,      880k used,  1019236k free,    95772k cached
PID to renice: 10604

10604 root      16   0  5396 1544 1244 S  0.0  0.4   0:00.07 bash

# 之后，可以输入 nice 值了！
top - 13:53:00 up 51 days,  2:27,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:   1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie
```

```
Cpu(s):  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:    385676k total,   371760k used,    13916k free,   131164k buffers
Swap:  1020116k total,      880k used,  1019236k free,    95772k cached
Renice PID 10604 to value: 10

10604 root      30  10  5396 1544 1244 S  0.0  0.4   0:00.07 bash
```

top 也是个挺不错的程序观察工具！但不同于 ps 是静态的结果输出， top 这个程序可以持续的监测 (monitor) 整个系统的程序工作状态，例如上面的范例一所示啊！ 在预设的情况下，每次更新程序资源的时间为 5 秒，不过，可以使用 -d 来进行修改。

top 主要分为两个画面，上面的画面为整个系统的资源使用状态，基本上总共有六行， 显示的内容依序是：

- 第一行：显示系统已启动的时间、目前上线人数、系统整体的负载(load)。 比较需要注意的是系统的负载，三个数据分别代表 1, 5, 10 分钟的平均负载。 一般来说，这个负载值应该不太可能超过 1 才对，除非您的系统很忙碌。 如果持续高于 5 的话，那么.....仔细的看看到底是那个程序在影响整体系统吧！
- 第二行：显示的是目前的观察程序数量，比较需要注意的是最后的 zombie 那个数值，如果不是 0 ，嘿嘿！好好看看到底是那个 process 变成疆尸了吧？！
- 第三行：显示的是 CPU 的整体负载，每个项目可使用 ? 查阅。需要观察的是 id (idle) 的数值，一般来说，他应该要接近 100% 才好，表示系统很少资源被使用啊！ ^_^。
- 第四行与第五行：表示目前的物理内存与虚拟内存 (Mem/Swap) 的使用情况。
- 第六行：这个是当在 top 程序当中输入指令时，显示状态的地方。 例如范例四就是一个简单的使用例子。

至于 top 底下的画面，则是每个 process 使用的资源情况。比较需要注意的是：

- PID ：每个 process 的 ID 啦！
- USER：该 process 所属的使用者；
- PR ：Priority 的简写，程序的优先执行顺序，越小越早被执行；
- NI ：Nice 的简写，与 Priority 有关，也是越小越早被执行；
- %CPU：CPU 的使用率；
- %MEM：内存的使用率；
- TIME+：CPU 使用时间的累加；

一般来说，如果鸟哥想要找出最损耗 CPU 资源的那个程序时，大多使用的就是 top 这支程序啦！然后强制以 CPU 使用资源来排序 (在 top 当中按下 P 即可)， 就可以很快的知道啦！ ^_^。多多爱用这个好用的东西喔！

- pstree

```
[root@linux ~]# pstree [-Aup]
参数：
-A  ：各程序树之间的连接以 ASCII 字符来连接；
```

```
-p  ：并同时列出每个 process 的 PID；
-u  ：并同时列出每个 process 的所属账号名称。
范例：

范例一：列出目前系统上面所有的程序树的相关性：
[root@linux ~]# pstree -A
init-+-atd
     |-crond
     |-dhclient
     |-dovecot-+-dovecot-auth
     |         `-3*[pop3-login]
     |-events/0
     |-2*[gconfd-2]
     |-master-+-pickup
     |        `-qmgr
     |-6*[mingetty]
     |-sshd---sshd---sshd---bash---su---bash---pstree
     |-udevd
     `-xinetd
# 注意一下，为了节省版面，所以鸟哥已经删去很多程序了！
# 同时注意到 sshd--- 那一行，嘿嘿！有相关的程序都被列出在一起了！

范例二：承上题，同时秀出 PID 与 users
[root@linux ~]# pstree -Aup
init(1)-+-atd(16143)
        |-crond(1676)
        |-dhclient(21339)
        |-dovecot(1606)-+-dovecot-auth(1616)
        |               |-pop3-login(747,dovecot)
        |               |-pop3-login(10487,dovecot)
        |               `-pop3-login(10492,dovecot)
        |-events/0(3)
        |-gconfd-2(2352)
        |-gconfd-2(32158)
        |-master(1666)-+-pickup(10817,postfix)
        |              `-qmgr(1675,postfix)
        |-mingetty(1792)
        |-mingetty(21366)
        |-sshd(5281)---sshd(10576)---sshd(10578,vbird)---bash(10579)
        |-syslogd(1539)
        |-udevd(801)
        `-xinetd(1589)
# 呵呵！在括号 () 内的即是 PID 以及该程序的 owner 喔！不过，由于我是使用
```

```
# root 的身份执行此一指令，所以啰，嘿嘿！属于 root 的可能就不会显示出来啦！
```

如果要找程序之间的相关性，呵呵！这个 pstree 真是好用到不行！ 直接输入 pstree 可以查到程序相关性，不过，有的时候由于语系的问题会出现乱码，因此，建议直接使用 -A 用 ASCII 字符作为连结接口 (就是那个 +, -, |, ` 等等啦！) 会比较看的清楚点。另外，如果还想要知道 PID 与所属使用者，加上 -u 及 -p 两个参数即可。 我们前面不是一直提到，如果子程序挂点或者是老是砍不掉子程序时， 该如何找到父程序吗？呵呵！用这个 pstree 就对了！ ^_^

程序的删除

我们在前几个小节提到的『背景工作管理』当中提到过， 要给予某个已经存在的工作某些动作时，是直接给予一个讯号 (signal) 给该 PID 即可。 常见的工作可以使用 kill -l (L 的小写) 来查阅！而主要的讯号代号与名称对应及内容是：

代号	名称	内容
1	SIGHUP	代表『让该 PID 重新读取自己的设定档』 ，类似重新启动
2	SIGINT	代表用键盘输入的 [ctrl]-c 来中断一个程序的进行。
9	SIGKILL	代表强制中断一个程序的进行，如果该程序进行到一半， 那么尚未完成的部分可能会有『半产品』产生，类似 vim 会有 .filename.swp 保留下来。
15	SIGTERM	以正常的结束程序来终止该程序。由于是正常的终止， 所以后续的动作会将他完成。不过，如果该程序已经发生问题，就是无法使用正常的方法终止时， 输入这个 signal 也是没有用的。

而 kill 可以帮我们将这个 signal 传送给某个工作 (%jobnumber) 或者是某个 PID (直接输入数字)，也就是说， kill 后面直接加数字与加上 % 的情况是不同的！这个很重要喔！不要搞错了。 我们就活用一下 kill 与刚刚上面提到的 ps 来做个简单的练习吧！

例题：以 ps 找出 syslog 这个服务的 PID 后，再使用 kill 重新读取 syslog 的设定文件数据：
答：
　　我们可以使用底下的方式找出 syslog 的 PID 喔！
　　　　ps aux | grep 'syslog' | grep -v 'grep'| awk '{print $2}'
　　接下来，再给予 kill -SIGHUP 的讯号至该 PID ，所以，整个指令串可以这样写：
　　　　kill -SIGHUP `ps aux|grep 'syslog'|grep -v 'grep'|awk '{print $2}'`
　　然后立刻 tail -n 5 /var/log/messages 看看 syslog 有没有重新被启动啊？

由于 kill 后面必须要加上 PID (或者是 job number)，所以，通常 kill 都会配合 ps, pstree 等指令，因为我们必须要找到相对应的那个程序的 ID 嘛！但是，如此一来，很麻烦～ 有没有可以利用『下达指令的名称』来给予讯号的？举例来说，能不能直接将 syslog 这个程序给予一个 SIGHUP 的讯号呢？可以的！用 killall 吧！

```
[root@linux ~]# killall [-iIe] [command name]
```

```
参数：
-i  ：interactive 的意思，交互式的，若需要删除时，会出现提示字符给使用者；
-e  ：exact 的意思，表示『后面接的 command name 要一致』，但整个完整的指令
      不能超过 15 个字符。
-I  ：指令名称(可能含参数)忽略大小写。
范例：

范例一：给予 syslogd 这个指令启动的 PID 一个 SIGHUP 的讯号
[root@linux ~]# killall -1 syslogd
# 如果用 ps aux 仔细看一下，syslogd 才是完整的指令名称。但若包含整个参数，
# 则 syslogd -m 0 才是完整的呢！

范例二：强制终止所有以 httpd 启动的程序
[root@linux ~]# killall -9 httpd
```

总之，要删除某个程序，我们可以使用 PID 或者是启动该程序的指令名称， 而如果要删除某个服务呢？呵呵！最简单的方法就是利用 killall ， 因为他可以将系统当中所有以某个指令名称启动的程序全部删除。 举例来说，上面的范例二当中，系统内所有以 httpd 启动的程序，就会通通的被删除啦！ ^_^

系统资源的观察

除了系统的程序之外，我们还必须就系统的一些资源进行检查啊！ 举例来说，我们使用 top 可以看到很多系统的资源对吧！那么，还有没有其它的工具可以查阅的？ 当然有啊！底下这些工具指令可以玩一玩！

- free

```
[root@linux ~]# free [-b|-k|-m|-g] [-t]
参数：
-b  ：直接输入 free 时，显示的单位是 Kbytes，我们可以使用 b(bytes), m(Mbytes)
      k(Kbytes), 及 g(Gbytes) 来显示单位喔！
-t  ：在输出的最终结果，显示物理内存与 swap 的总量。
范例：

范例一：显示目前系统的内存容量
[root@linux ~]# free -m
             total       used       free     shared    buffers     cached
Mem:           376        366         10          0        129         94
-/+ buffers/cache:        141        235
Swap:          996          0        995
```

仔细看看，我的系统当中有 384 MB 左右的物理内存，我的 swap 有 1GB 左右， 那我使用 free -m 以 MBytes 来显示时，就会出现上面的信息。Mem 那一行显示的是物理内存的量， Swap 则是虚拟内存的量。 total 是总量， used 是已被使用的量， free 则是剩余可用的量。 后面的 shared/buffers/cached 则是在已被使用的量当中，用来作为缓冲及快取的量。

仔细的看到范例一的输出喔，我的 Linux 主机是很平凡的，根本没有什么工作， 但是，我的物理内存是几乎被用光光的情况呢！不过，至少有 129 MB 用在缓冲记忆工作， 94 MB 则用在快取工作，也就是说，系统是『很有效率的将所有的内存用光光』， 目的是为了让系统的存取效能加速啦！

很多朋友都会问到这个问题『我的系统明明很轻松，为何内存会被用光光？』 现在瞭了吧？没有错！被用光是正常的！而需要注意的反而是 swap 的量。一般来说， swap 最好不要被使用，尤其 swap 最好不要被使用超过 20% 以上， 如果您发现 swap 的用量超过 20% ，那么，最好还是买物理内存来插吧！ 因为，Swap 的效能跟物理内存实在差很多，而系统会使用到 swap ， 绝对是因为物理内存不足了才会这样做的！如此，了解吧！

- uname

```
[root@linux ~]# uname [-asrmpi]
参数：
-a  ：所有系统相关的信息；
-s  ：系统核心名称
-r  ：核心的版本
-m  ：本系统的硬件名称
-p  ：CPU 的类型
-i  ：硬件的平台 (ix86)
范例：

范例一：输出系统的基本信息
[root@linux ~]# uname -a
Linux linux.site 2.6.12-1.1398_FC4 #1 Fri Jul 15 00:52:32 EDT 2005
i686 i686 i386 GNU/Linux
```

这个咚咚我们前面使用过很多次了喔！uname 可以列出目前系统的核心版本、 主要硬件平台以及 CPU 类型等等的信息。以上面范例一的状态来说，我的 Linux 主机使用的核心名称为 Linux，而主机名称为 linux.site，核心的版本为 2.6.12-1.1398_FC4，该核心版本建立的日期为 2005/07/15， 适用的硬件平台为 i386 以上等级的硬件平台喔。

- uptime

这个指令很单纯呢！就是显示出目前系统已经开机多久的时间，以及 1, 5, 15 分钟的平均负载就是了。还记得 top 吧？没错啦！ 这个 uptime 可以显示出 top 画面的最上面一行！

```
[root@linux ~]# uptime
 18:06:30 up 52 days,  6:40,  1 user,  load average: 0.00, 0.00, 0.00
# 上面表示，目前是 18:06 ，本系统已经开机 52 天又 6:40 ，有 1 个使用者在在线，
# 平均负载很低，所以都是 0 啊！
```

- netstat

这个 netstat 也是挺好玩的，其实，这个指令比较常被用在网络的监控方面， 不过，在程序管理方面也是需要了解的啦！这个指令的执行如下所示：基本上， netstat 的输出分为两大部分，上面是网络接口相关的联机，下方则是与 unix 程序有关的项目。

```
[root@linux ~]# netstat -[atunlp]
参数：
-a  ：将目前系统上所有的联机、监听、Socket 数据都列出来
-t  ：列出 tcp 网络封包的数据
-u  ：列出 udp 网络封包的数据
-n  ：不已程序的服务名称，以埠号 (port number) 来显示；
-l  ：列出目前正在网络监听 (listen) 的服务；
-p  ：列出该网络服务的程序 PID
范例：

范例一：列出目前系统已经建立的网络联机与 unix socket 状态
[root@linux ~]# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address         Foreign Address       State
tcp        0    256 59-125-83-224.ad:ssh linux.test.s:52679    ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  16     [ ]         DGRAM                    4870   /dev/log
unix  2      [ ]         DGRAM                    3561   @udevd
unix  3      [ ]         STREAM     CONNECTED     509237
# 在上面的结果当中，显示了两个部分，分别是网络的联机以及 linux 上面的 socket
# 联机状态。在网络联机的部分主要内容为：
# Proto ：网络的封包协议，主要分为 TCP 与 UDP 封包，相关数据请参考服务器篇；
# Recv-Q：非由使用者程序连结到此 socket 的复制的总 bytes 数；
# Send-Q：非由远程主机传送过来的 acknowledged 总 bytes 数；
# Local Address  ：本地端的 IP
# Foreign Address：远程主机的 IP；
# State ：联机状态，主要有建立(ESTABLISED)及监听(LISTEN)；
# 至于 unix 传统的 socket 连接状态则是：
# Proto ：一般就是 unix 啦；
# RefCnt：连接到此 socket 的程序数量；
# Flags ：联机的旗标；
# Type  ：socket 存取的类型。主要有确认联机的 STREM 与不需确认的 DGRAM 两种；
# State ：CONNECTED 表示已经联机建立。
# Path  ：连接到此 socket 的相关程序的路径！或者是相关数据输出的路径。

范例二：找出目前系统上已在监听的网络联机及其 PID
[root@linux ~]# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address    Foreign Address  State    PID/Program name
```

```
tcp        0      0 0.0.0.0:21       0.0.0.0:*         LISTEN   1598/vsftpd
tcp        0      0 127.0.0.1:25     0.0.0.0:*         LISTEN   1666/master
tcp        0      0 :::22            :::*              LISTEN   5281/sshd
udp        0      0 0.0.0.0:68       0.0.0.0:*                  21339/dhclient
# 看到了吗？最后面一个字段就是该埠号被该 PID 或程序所启动的！

范例三：将上述的本地端 0.0.0.0:21 那个网络服务关闭的话？
[root@linux ~]# kill 1598
[root@linux ~]# killall vsftpd
```

很多朋友常常有疑问，那就是，我的主机目前到底开了几个门(ports)， 呵呵！其实，不论主机提供什么样的服务，一定必须要有相对应的 program 在主机上面执行才行啊！ 举例来说，我们鸟园的 Linux 主机提供的就是 WWW 服务，那么我的主机当然有一个程序在提供 WWW 的服务啊！呵呵！那就是 Apache 这个套件所提供的啦！ ^_^。 所以，当我执行了这个程序之后，我的系统自然就可以提供 WWW 的服务了。那如何关闭啊？ 就关掉该程序所触发的那个程序就好了！例如上面的范例三所提供的例子啊！ ^_^

- dmesg

在开机的时候你会发现有很多的讯息出现吧，例如 CPU 的形式、硬盘、 光盘型号及硬盘分割表等等，这些信息的产生都是核心 (kernel) 在进行硬件的测试与驱动啦。 但是讯息都是『刷』的一声就跑过去了！完全来不及看！伤脑筋～

这些讯息有时候对于系统管理员是很重要的，因为他提供了系统的信息呀！ 要看这些讯息你可以用 dmesg 这个指令来观看！ 因为讯息实在太多了，所以可以加入这个管线指令『 | more 』来使画面暂停！

```
范例一：输出所有的核心开机时的信息
[root@linux ~]# dmesg | more

范例二：搜寻开机的时候，硬盘的相关信息为何？
[root@linux ~]# dmesg | grep -i hd
    ide0: BM-DMA at 0xffa0-0xffa7, BIOS settings: hda:DMA, hdb:DMA
    ide1: BM-DMA at 0xffa8-0xffaf, BIOS settings: hdc:DMA, hdd:pio
hda: ST320430A, ATA DISK drive
hdb: Maxtor 5T030H3, ATA DISK drive
hdc: CD-540E, ATAPI CD/DVD-ROM drive
.....底下省略.....
```

由范例二就知道我这部主机的硬盘是怎样了吧？！没错啦！ 还可以查阅能不能找到网络卡喔！网络卡的代号是 eth ，所以，直接输入 dmesg | grep -i eth 试看看呢！

- sar

这个 sar 并不是系统预设的安装套件，如果您不是选择全部安装的话，这个套件预设是不装的。 不过，如果您是选择全部安装，嘿嘿！那就可以玩这个 sar 了。 这个 sar 的功能倒是可以玩一玩的，因为他可以在您想要主动侦测主机的资源状态， 然后绘制成为图表时，相当好用的一个工具喔！

```
[root@linux ~]# sar [-ru] [秒数] [次数]
参数：
-u  ：进行 CPU 资源的统计；
-r  ：进行主存储器目前状态的分析
范例：

范例一：统计目前主机 CPU 状态，每秒一次，共计三次！
[root@linux ~]# sar -u 1 3
Linux 2.6.12-1.1398_FC4 (vbird.vbird.idv.tw)    09/16/05

14:16:17          CPU     %user     %nice   %system   %iowait     %idle
14:16:18          all      0.00      0.00      0.00      0.00    100.00
14:16:19          all      0.00      0.00      0.00      0.00    100.00
14:16:20          all      0.00      0.00      0.00      0.00    100.00
Average:          all      0.00      0.00      0.00      0.00    100.00
# 我这部主机单纯用在家里测试的，所以没有什么网络服务，看的出来，嘿嘿！很安静！

范例二：统计目前主机内存的使用情况
[root@linux ~]# sar -r 1 3
Linux 2.6.12-1.1398_FC4 (vbird.vbird.idv.tw)    09/16/05

14:17:40    kbmemfree kbmemused  %memused kbbuffers  kbcached kbswpfree
14:17:41        26004    359672     93.26    127528     83996   1019236
14:17:42        26004    359672     93.26    127528     83996   1019236
14:17:43        26004    359672     93.26    127528     83996   1019236
Average:        26004    359672     93.26    127528     83996   1019236
# 其实这个与 free 的输出结果也差不了太多啦！
```

鸟哥倒是很喜欢使用 sar 来做背景主动侦测系统 CPU 的动作！参考看看先！

关于程序的执行顺序：

还记得我们提过的多人多任务环境吧？因为目前的 x86 平台的 CPU 可以做到多任务的行为， 所以啰，我们的 Linux 可以在 x86 上面『同时进行多个工作』的呢！那么多个工作是如何进行的呢？ 其实每个工作都会进入到 CPU 的工作排程当中，并等待 CPU 来执行， 而 CPU 会根据每个工作的优先执行序 (priority) 来判断谁比较重要， 所以某个工作就可能会比较优先被执行完毕啦！

也就是说， Linux 系统中，每个 process 都会拥有一个所谓的『优先执行序 (priority)』的属性， 利用该属性来让 CPU 判断那个工作是比较重要的，那个工作在一群工作当中就会优先被执行， 也让系统资源可以分配的更恰当。我们可以使用 ps 还观察优先执行序：

```
[root@linux ~]# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0 18851 18827  0  77   0 -  1302 wait   pts/0    00:00:00 su
4 S     0 18852 18851  0  76   0 -  1349 wait   pts/0    00:00:00 bash
```

```
4 R     0 19510 18852  0  76    0 -  1111 -      pts/0    00:00:00 ps
```

其中,那个 PRI 就是 Priority 的简写,而 NI 是 nice 的简写,这两个东西是凑在一起才产生目前的 PRI 值的! PRI 越小时,代表该程序可以具有『越早被优先执行』的意思,只是 PRI 是由系统动态产生的, 并不会是一直固定的值喔。至于那个 NI (nice) 则是我们操作值额外给予的一个数值, 他可以影响 PRI 的值,基本上,他的相关性是这样的:

- PRI(new) = PRI(old) + nice

不过您要特别留意到,如果原本的 PRI 是 50 ,并不是我们给予一个 nice = 5 ,就会让 PRI 变成 55 喔! 因为 PRI 是系统『动态』决定的,所以,虽然 nice 值是可以影响 PRI ,不过, 最终的 PRI 仍是要经过系统分析后才会决定的。另外, nice 值是有正负的喔,而既然 PRI 越小越早被执行, 所以,当 nice 值为负值时,那么该程序就会降低 PRI 值,亦即会变的较优先被处理。此外,您必须要留意到:

- 一般使用者的 nice 值为 0 ~ 19 ;
- root 可用的 nice 值为 -20 ~ 19 ;
- 一般使用者仅可将 nice 值越调越高,如果本来 nice 为 5 ,则未来仅能调整到大于 5 的 nice ;
- 一般使用者仅能调整属于自己的程序的 nice 值。

这也就是说,要调整某个程序的优先执行序,就是『调整该程序的 nice 值』啦!那么如何给予某个程序 nice 值呢?有两种方式,分别是:

- 一开始执行程序就立即给予一个特定的 nice 值:用 nice 指令;
- 调整某个已经存在的 PID 的 nice 值:用 renice 指令。

- nice

```
[root@linux ~]# nice [-n] command
参数:
-n  : 后面接一个数值,数值的范围 -20 ~ 19。
范例:

范例一:用 root 给一个 nice 植为 -5 ,用于执行 vi ,并观察该程序!
[root@linux ~]# nice -n -5 vi &
[1] 19542
[root@linux ~]# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0 18851 18827  0  77    0 -  1302 wait   pts/0    00:00:00 su
4 S     0 18852 18851  0  76    0 -  1349 wait   pts/0    00:00:00 bash
4 T     0 19542 18852  0  72   -5 -  1063 finish pts/0    00:00:00 vi
4 R     0 19543 18852  0  77    0 -  1110 -      pts/0    00:00:00 ps
```

就如同前面说的， nice 是用来调整程序的执行优先级！这里只是一个执行的范例罢了！ 通常什么时候要将 nice 值调大呢？举例来说，系统的背景工作中， 某些比较不重要的程序之进行：例如备份工作！由于备份工作相当的耗系统资源， 这个时候就可以将备份的指令之 nice 值调大一些，可以使系统的支持分配的更为公平！

- renice

```
[root@linux ~]# renice [number] PID
参数：
PID ：某个程序的 ID 啊！
范例：

范例一：以上面 nice 范例中 ps -l 的结果，将 18852 那个 PID 修改 nice 为 10
[root@linux ~]# renice 10 18852
18852: old priority 0, new priority 10
[root@linux ~]# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0 18851 18827  0  77   0 -  1302 wait   pts/0    00:00:00 su
4 S     0 18852 18851  0  85  10 -  1349 wait   pts/0    00:00:00 bash
4 R     0 19593 18852  0  87  10 -  1111 -      pts/0    00:00:00 ps
```

如果要调整的是已经存在的某个 process 的话，那么就得要使用 renice 了。使用的方法很简单， renice 后面接上数值及 PID 即可。因为后面接的是 PID ，所以您务必要以 ps 或者其它程序观察的指令去找出 PID 才行啊！

刚好，由上面这个范例当中我们也看的出来，虽然修改的是 bash 那个 PID 为 18852 的程序， 但是该程序所触发的 ps 指令当中的 PID 同样的也有一个 nice = 10 的结果喔！ 了解了吧？整个 nice 值是可以在父程序 --> 子程序之间传递的呢！

另外，除了 renice 之外，其实那个 top 同样的也是可以调整 nice 值的！ top 也是可以调整已经存在的某个 process 的 nice 喔！

特殊档案与程序：

我们在 档案与目录管理当中的 SUID/SGID/SBIT 提到这一些奇怪的特殊权限的档案，这些档案到底是怎么产生特殊权限的行程？还有， 在磁盘挂载的时候，有时老是出现『 device is busy 』的字样，真麻烦～到底是怎么回事啊？ 我们底下就来谈一谈。

SUID/SGID/SBIT 的概念

虽然已经在前面的章节提过 SUID/SGID 等等的特殊权限，但是到底他是如何产生的？ 首先，我们以条列式的方法列出使用者是否能够操作 SUID/SGID 的指令吧！

- 关于 SUID 的执行：

- Set UID 的权限设定值，仅对 binary file 有效；
- 程序操作者必须要拥有该 binary file 的可执行权限 (x) ；
- 当程序操作者执行该具有 SUID 的 binary file 时，该 binary file 所触发的程序中，该程序的有效使用者 (effective user) 为该 binary file 的拥有者。

很简单的概念吧！基本上，也就是说，使用者所触发的 SUID 的程序中， 有效使用者会变成该 SUID 指令的程序拥有者啦！我们以 pstree 配合 passwd 来做个实验好咯。 请您以 dmtsai 这个一般身份使用者登入后，执行 passwd ，但是先不要输入密码， 然后以另外一个终端机登入系统，使用 pstree -u 后，应该会看到如下的画面喔：

```
init-+-atd
     |-.....省略的啦.........
     |-sshd---sshd---bash(dmtsai)---passwd(root)
     |-.....省略的啦.........
     `-xinetd
```

看到了吧？虽然我是以 dmtsai 的身份启动 bash ，然后在 bash 当中执行 /usr/bin/passwd， 不过由于 passwd 拥有 SUID 的权限设定，所以，在 run-time 的过程当中，嘿嘿！ 我这个 dmtsai 在该 process 内可是拥有 /usr/bin/passwd 的 owner (亦即是 root) 的权限喔！这样够明白了吧？！

那么既然 SUID/SGID 的权限是比较可怕的，您该如何查询整个系统的 SUID/SGID 的档案呢？ 应该是还不会忘记吧？使用 find 即可啊！

```
find / -perm +6000
```

/proc/* 代表的意义：

其实，我们之前提到的所谓的程序都是在内存当中嘛！而内存当中的数据又都是写入到 /proc/* 这个目录下的，所以啰，我们当然可以直接观察 /proc 这个目录当中的档案啊！ 如果您观察过 /proc 这个目录的话，应该会发现他有点像这样：

```
[root@linux ~]# ll /proc
dr-xr-xr-x   5 root    root           0 Sep 12 14:12 1
dr-xr-xr-x   5 root    root           0 Sep 15 12:01 10
dr-xr-xr-x   5 dovecot dovecot        0 Sep 14 12:07 10487
.....中间省略.....
-r--r--r--   1 root    root           0 Sep 16 16:02 uptime
-r--r--r--   1 root    root           0 Sep 16 16:02 version
-r--r--r--   1 root    root           0 Sep 16 16:02 vmstat
```

基本上，目前主机上面的各个程序的 PID 都是以目录的型态存在于 /proc 当中。 举例来说，我们开机所执行的第一支程序 init 他的 PID 是 1 ， 这个 PID 的所有相关信息都写入在 /proc/1/* 当中！若我们直接观察 PID 为 1 的数据好了， 他有点像这样：

```
[root@linux ~]# ll /proc/1
dr-xr-xr-x  2 root root 0 Sep 16 16:04 attr
-r--------  1 root root 0 Sep 16 16:04 auxv
-r--r--r--  1 root root 0 Sep 16 10:23 cmdline
```

```
lrwxrwxrwx  1 root root 0 Sep 16 10:23 cwd -> /
-r--------  1 root root 0 Sep 16 16:04 environ
lrwxrwxrwx  1 root root 0 Sep 16 10:23 exe -> /sbin/init
.....以下省略.....
```

里面的数据还挺多的，不过，比较有趣的其实是两个档案，分别是：

- cmdline：这个程序被启动的指令串；
- environ：这个程序的环境变量内容。

很有趣吧！如果你查阅一下 cmdline 的话，就会发现：

```
[root@linux ~]# cat /proc/1/cmdline
init [3]
```

就是这个指令与参数启动 init 的啦!这还是跟某个特定的 PID 有关的内容呢,如果是针对整个 Linux 系统相关的参数呢？那就是在 /proc 目录底下的档案啦！相关的档案与对应的内容是这样的：

檔名	档案内容
/proc/cmdline	加载 kernel 时所下达的相关参数！查阅此档案，可了解系统是如何启动的！
/proc/cpuinfo	本机的 CPU 的相关信息，包含频率、类型与运算功能等
/proc/devices	这个档案记录了系统各个主要装置的主要装置代号，与 mknod 有关呢！
/proc/filesystems	目前系统已经加载的档案系统啰！
/proc/interrupts	目前系统上面的 IRQ 分配状态。
/proc/ioports	目前系统上面各个装置所配置的 I/O 地址。
/proc/kcore	这个就是内存的大小啦！好大对吧！但是不要读他啦！
/proc/loadavg	还记得 top 以及 uptime 吧？没错！上头的三个平均数值就是记录在此！
/proc/meminfo	使用 free 列出的内存信息，嘿嘿！在这里也能够查阅到！
/proc/modules	目前我们的 Linux 已经加载的模块列表，也可以想成是驱动程序啦！
/proc/mounts	系统已经挂载的数据，就是用 mount 这个指令呼叫出来的数据啦！
/proc/swaps	到底系统挂加载的内存在哪里？呵呵！使用掉的 partition 就记录在此啦！
/proc/partitions	使用 fdisk -l 会出现目前所有的 partition 吧？在这个档案当中也有纪录喔！
/proc/pci	在 PCI 总线上面，每个装置的详细情况！可用 lspci 来查阅！
/proc/uptime	就是用 uptime 的时候，会出现的信息啦！
/proc/version	核心的版本，就是用 uname -a 显示的内容啦！
/proc/bus/*	一些总线的装置，还有 USB 的装置也记录在此喔！

其实，上面这些档案鸟哥在此建议您可以使用 cat 去查阅看看，不必深入了解， 不过，观看过档案内容

后，毕竟会比较有感觉啦！如果未来您想要自行撰写某些工具软件， 那么这个目录底下的相关档案可能会对您有点帮助的喔！

查询已开启档案或已执行程序开启之档案：

其实还有一些与程序相关的指令可以值得参考与应用的，我们来谈一谈：

- fuser

如果当我们要卸载某个装置时，他老是告诉我们『 device is busy 』， 那么到底是那个程序在使用这个装置呢？举例来说，当无法 umount /home 时， 该怎么办？此时我们可以使用 fuser 来帮忙啦！

```
[root@linux ~]# fuser [-ki] [-signal] file/dir
参数：
-k  ：找出使用该档案/目录的 PID ，并试图以 SIGKILL 这个讯号给予该 PID；
-i  ：必须与 -k 配合，在删除 PID 之前会先询问使用者意愿！
-signal：例如 -1 -15 等等，若不加的话，预设是 SIGKILL (-9) 啰！
范例：

范例一：找出目前所在目录的使用 PID 为何？
[root@linux ~]# fuser .
.:                   18852c
[root@linux ~]# ps aux | grep 18852
root     18852  0.0  0.4   5396  1588 pts/0    SN   10:12   0:00 bash
# 用这个方式就可以得到使用该目录的 PID 了。此外，为何使用 fuser
# 的输出当中，在 PID 后面会有 c 呢？他代表的意义为：
# c ：在当前的目录下；
# e ：可以被执行的；
# f ：是一个被开启的档案
# r ：代表 root directory

范例二：找到 /var 底下属于 FIFO 类型的档案，并且找出存取该档案的程序
[root@linux ~]# find /var -type p
/var/spool/postfix/public/qmgr
/var/spool/postfix/public/pickup
[root@linux ~]# fuser /var/spool/postfix/public/qmgr
/var/spool/postfix/public/qmgr:  1666  1675
[root@linux ~]# ps aux | egrep '(1666|1675)'
root    1666 0.0 0.3 5640 1516 ?  Ss Jul25  0:01 /usr/libexec/postfix/master
postfix 1675 0.0 0.4 5744 1604 ?  S  Jul25  0:00 qmgr -l -t fifo -u

范例三：同范例二，但试图删除该 PID?
[root@linux ~]# fuser -ki /var/spool/postfix/public/qmgr
/var/spool/postfix/public/qmgr:  1666  1675
Kill process 1666 ? (y/N) n
```

```
Kill process 1675 ? (y/N) n
```

如何？很有趣的一个指令吧！透过这个 fuser 我们可以找出使用该档案、 目录的程序，藉以观察的啦！

- lsof

相对于 fuser 是由档案或者装置去找出使用该档案或装置的程序，反过来说， 如何查出某个程序开启或者使用的档案与装置呢？呼呼！那就是使用 lsof 啰～

```
[root@linux ~]# lsof [-Uu] [+d]
参数：
-a  ：多项数据需要『同时成立』才显示出结果时！
-U  ：仅列出 Unix like 系统的 socket 档案类型；
-u  ：后面接 username，列出该使用者相关程序所开启的档案；
+d  ：后面接目录，亦即找出某个目录底下已经被开启的档案！
范例：

范例一：列出目前系统上面所有已经被开启的档案与装置：
[root@linux ~]# lsof
COMMAND PID USER   FD   TYPE  DEVICE   SIZE    NODE NAME
init      1 root cwd    DIR     3,1    4096       2 /
init      1 root rtd    DIR     3,1    4096       2 /
init      1 root txt    REG     3,1   34352  883193 /sbin/init
.....底下省略.....
# 注意到了吗？是的，在预设的情况下， lsof 会将目前系统上面已经开启的
# 档案全部列出来～所以，画面多的吓人啊！您可以注意到，第一个档案 init 执行的
# 地方就在根目录，而根目录，嘿嘿！所在的 inode 也有显示出来喔！

范例二：仅列出关于 root 的所有程序开启的 socket 档案
[root@linux ~]# lsof -u root -a -U
COMMAND     PID USER   FD   TYPE      DEVICE SIZE   NODE NAME
kmodule     793 root    4u  unix 0xd744b700         3549 socket
udevd       801 root    5u  unix 0xd744bb40         3561 socket
syslogd    1539 root    0u  unix 0xd75946e0         4870 /dev/log
# 注意到那个 -a 吧！如果你分别输入 lsof -u root 及 lsof -U ，会有啥信息？
# 使用 lsof -u root -U 及 lsof -u root -a -U ，呵呵！都不同啦！
# -a 的用途就是在解决同时需要两个项目都成立时啊！ ^_^

范例三：请列出目前系统上面所有的被启动的周边装置
[root@linux ~]# lsof +d /dev
COMMAND     PID    USER   FD   TYPE     DEVICE SIZE NODE NAME
init          1    root   10u  FIFO       0,13      1834 /dev/initctl
kmodule     793    root    2u   CHR        1,3      2135 /dev/null
kmodule     793    root    3u   CHR        5,1      2134 /dev/console
udevd       801    root    2u   CHR        1,3      2135 /dev/null
syslogd    1539    root    0u  unix 0xd75946e0      4870 /dev/log
```

```
xinetd     1589    root    1r   CHR         1,3       2135 /dev/null
#看吧！因为装置都在 /dev 里面嘛！所以啰，使用搜寻目录即可啊！

范例四：秀出属于 root 的 bash 这支程序所开启的档案
[root@linux ~]# lsof -u root | grep bash
bash   26199 root  cwd    DIR   3,2    4096   159875 /root
bash   26199 root  rtd    DIR   3,1    4096        2 /
bash   26199 root  txt    REG   3,1 686520   294425 /bin/bash
bash   26199 root  mem    REG   3,1  83160    32932 /usr/lib/gconv/BIG5.so
bash   26199 root  mem    REG   3,1  46552   915764 /lib/libnss_files-2.3.5.so
.....底下省略.....
```

这个指令可以找出您想要知道的某个程序是否有启用哪些信息？ 例如上头提到的范例四的执行结果呢！^_^

- pidof

```
[root@linux ~]# pidof [-sx] program_name
参数：
-s  ：仅列出一个 PID 而不列出所有的 PID
-x  ：同时列出该 program name 可能的 PPID 那个程序的 PID
范例：

范例一：列出目前系统上面 init 以及 syslogd 这两个程序的 PID
[root@linux ~]# pidof init syslogd
1 2546
# 理论上，应该会有两个 PID 才对。上面的显示也是出现了两个 PID 喔。
# 分别是 init 及 syslogd 这两支程序的 PID 啦。

范例二：找出 bash 即以 bash 为 PPID 的几个主要的 PID
[root@linux ~]# pidof -x bash
2961 2959 338
# 因为我的系统被我登入之后，我就会主动取得一个 bash 的程序，所以啰，
# 很自然就会拥有一个 PID 啊。只要我再以底下的方式，就可以取得我所想要的 PID 内容。
[root@linux ~]# ps aux | egrep '(2961|2959|338)'
dmtsai    338  0.0  0.1  6024 1536 pts/0    Ss   16:43   0:00 -bash
kiki     2961  0.0  0.1  6025 1526 pts/0    Ss   17:43   0:00 -bash
.....以下省略......
```

很简单的用法吧，透过这个 pidof 指令，并且配合 ps aux 与正规表示法， 就可以很轻易的找到您所想要的程序内容了呢。

参考数据

- 来自 Linux Journal 的关于 /proc 的说明：http://www.linuxjournal.com/article/177

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 简单说明什么是程序 (program) 而什么是程序 (process)？

  程序 (program) 是系统上面可以被执行的档案，由于 Linux 的完整檔名 (由 / 写起) 仅能有一个， 所以 program 的档名具有单一性。当程序被执行后，就会启动成程序 (process)， 一个 program 可以被不同的使用者或者相同的使用者重复的执行成为多个程序， 且该程序所造成的程序还因为不同的使用者，而有不同的权限，且每个 process 几乎都是独立的。

- 我今天想要查询 /etc/crontab 与 crontab 这个程序的用法与写法，请问我该如何在线查询？

  查询 crontab 指令可以使用 man crontab 或 info crontab ，至于查询 /etc/crontab ，则可以使用 man 5 crontab 啰！

- 我要如何查询 crond 这个 daemon 的 PID 与他的 PRI 值呢？

  ps -aux | grep crond 即可查到！

- 我要如何修改 crond 这个 PID 的优先执行序？

  先以 ps -aux 找到 crond 的 PID 后，再以： renice -n number PID 来调整！

- 我是一般身份使用者，我是否可以调整不属于我的程序的 nice 值？此外，如果我调整了我自己的程序的 nice 值到 10 ，是否可以将他调回 5 呢？

  不行！一般身份使用者仅能调整属于自己的 PID 程序，并且，只能将 nice 值一再地调高，并不能调低，所以调整为 10 之后，就不能降回 5 啰！

- 我要怎么知道我的网络卡在开机的过程中有没有被捉到？

  可以使用 dmesg 来视察！

开机关机流程与 Loader

最近更新日期：2005/11/09

在这个章节当中，我们特别要来看一看整个开机的流程设定， 看看能不能在开机的时候就主动的帮我们将所需要的信息都填进去！此外，还想要知道一下， 我们要如何来设定多重开机呢？设定多重开机的原理是什么？最重要的是那个『什么是开机管理程序 (boot loader) 呢？』这些东西对于家里只有一部计算机，却又要安装多个操作系统的朋友来说， 是相当重要而有趣的项目呢！鸟哥底下会介绍 Linux 下的两套相当棒的 boot loader 系统，分别是 lilo 及 grub。

开机流程简介

开机不是只要按一下电源钮而关机只要关掉电源钮就可以了吗？有何大学问？话是这样没错啦，但是由于 Linux 是一套多人多任务的操作系统，你难保你在关机时没有人在在线，如果你关机的时候碰巧一大群人在在线工作， 那会让当时在在线工作的人马上断线的！那不是害死人了！一些数据可是无价之宝哩！

另外，与 DOS 环境不同的是， Linux 在执行的时候，虽然你在画面上只会看到黑压压的一片，完全没有任何画面， 但其实他是有很多的程序在背景底下执行的，例如登录文件管控程序、前面两章提到的例行性命令， 当然还有一大堆网络服务，如邮件服务器、WWW服务器等等。你如果随便关机的话， 是很容易伤害硬盘及数据传输的动作的！所以在 Linux 下关机可是一门大学问喔。

既然开机是很严肃的一件事，呵呵，那我们来了解一下整个开机的过程吧！ 好让大家比较容易发现开机过程里面发生错误的地方，与解决之道！不过，由于开机的过程中，那个开机管理程序 (Boot Loader) 使用的软件可能不一样，例如目前各大 Linux distributions 的主流为 grub， 但早期 Linux 预设是使用 LILO ，台湾地区则很多朋友喜欢使用 spfdisk 。 但无论如何，我们总是得要了解整个 boot loader 的工作情况，才能了解为何进行多重开机的设定时， 为何老是听人家讲要先安装 Windows 再安装 Linux 的原因～

我们先来想一想， Linux 整个开机的程序是怎样呢？还记得我们提过，开机时要加载核心， 让核心来驱动整个硬件，这样才能算是一个最阳春、最基础的操作系统吧？然后才能够执行各种程序的运作。 同样的，开机的流程也是需要先加载核心的。不过，加载核心前，却需要一些前置作业，才能够正确无误的加载核心嘛！ 所以，整个开机的程序是这样的：

1. 加载 BIOS 的硬件信息，并取得第一个开机装置的代号；
2. 读取第一个开机装置的 MBR 的 boot Loader (亦即是 lilo, grub, spfdisk 等等) 的开机信息；
3. 加载 Kernel 操作系统核心信息， Kernel 开始解压缩，并且尝试驱动所有硬件装置；
4. Kernel 执行 init 程序并取得 run-level 信息；
5. init 执行 /etc/rc.d/rc.sysinit 档案；
6. 启动核心的外挂模块 (/etc/modprobe.conf)；
7. init 执行 run-level 的各个批次档(Scripts)；
8. init 执行 /etc/rc.d/rc.local 档案；
9. 执行 /bin/login 程序，并等待使用者登入；
10. 登入之后开始以 Shell 控管主机。

大概的流程就是上面写的那个样子啦，而每一个程序的内容主要是在干嘛呢？底下就分别来谈一谈吧！

---

boot loader 与 kernel 载入

由第一篇里面谈到的一些基础的主机硬件概念当中，我们知道整个主机在开机的时候，第一个被读取的地方， 就是 BIOS (Basic Input Output System) 啦，这个 BIOS 里面记录了主机板的芯片组与相关的设定， 例如 CPU 与接口设备的沟通频率啊、开机装置的搜寻顺序啊、硬盘的大小与类型啊、 系统时间啊、各周边总线的是否启动 Plug and Play (PnP, 随插即用装置) 啊、 各接口设备的 I/O 地址啊、以及与 CPU 沟通的 IRQ 岔断等等的信息都记录在此， 所以啰，系统要顺利的开机，首先就是要去读取 BIOS 的相关设定值了。

读取了 BIOS 设定值之后，系统会根据 BIOS 的数据，进行开机自我测试 (power on self test, POST)，

然后开始执行硬件侦测的初始化，并设定 PnP 装置，之后再定义出可开机的装置， 之后就会开始进行开机装置的数据读取了 (MBR 相关的任务开始)。

读完了 BIOS 并且了解了主要的主机硬件相关信息后，主机便会开始尝试由储存媒体加载操作系统了。 我们刚刚提到 BIOS 会记录『可用来开机的装置搜寻顺序』对吧！所以，系统会开始去第一个开机装置上面进行开机程序。 我们在第二篇的 磁盘档案系统(filesystem) 当中提到过整个储存装置的特性， 如果以硬盘来看，那么开机流程读到硬盘的过程中，第一个要读取的就是该硬盘的主要开机扇区 (Master Boot Record, MBR) 了，而系统可以由主要开机区所安装的开机管理程序 (boot loader) 开始执行核心辨识的工作。

Tips:
我们知道每颗硬盘的第一个扇区称为 MBR ，那么如果我的主机上面有两颗硬盘的话，系统会去哪颗硬盘的 MBR 读取数据呢？这个就得要看 BIOS 的设定了。 基本上，我们常常讲的『系统的 MBR』其实指的是 第一个开机装置的 MBR 才对！ 所以，改天如果您要将开机管理程序安装到某颗硬盘的 MBR 时， 要特别注意当时系统的『第一个开机装置』是那个，否则会安装到错误的硬盘上面喔！重要重要！



那么为什么要在 MBR 安装 boot loader 呢？而这个 boot loader 有什么功能呢？ 还记得我们在第二篇提到的 磁盘档案系统 吧？ 我们的操作系统核心必须要认识磁盘档案系统才能读取里面的数据啊， 但是整个系统才刚刚到开机起头的地方而已，要如何认识磁盘档案格式呢？ 那就得要藉由 boot loader 来辅助啦！所以啰，当然必须要有 boot loader 才有办法加载 Linux 的核心 (kernel) 啊！由于 boot loader 的特殊功能，因此，想要加载 Linux 核心时， 当然得使用支持 Linux filesystem 的 boot loader 了，目前主流的 grub 这套开机管理程序， 不但可以支持 Linux ，同时也支持 Windows 相关的核心系统呢！

好了，先再来回忆一下，如果你是以 grub 程序开机的话，那么在开机的时候会显示什么数据呢？呵呵！ 会显示蛮多的开机选单，没错～就是『选单』，然后选择了你的选择项目之后， 系统就会跑到该扇区去读取该操作系统的核心啰！呵呵！所以一个好的 boot loader 会具有两个功能，就是：

- 选单功能 (menu)
- 指向功能 (pointer)

再来强调一下，因为 Windows 与 Linux 的档案格式不一样？！ 为了加载系统核心，所以必须要安装认识我们操作系统的 loader， 而 Linux 的 loader (lilo 或 grub) 是可以认识 windows 的核心档案的，但是 Windows 的 loader 却不认识 Linux 的核心档案，因此，作为一个多重开机的设定 loader ，就无法使用 Windows 所提供的 loader 啰!由于需要让系统认识你的 kernel ，因此，就需要 boot loader 啦!这样想就对啦！

好了，当我们藉由 boot loader 的管理而开始读取核心档案后，接下来， Linux 就会将核心解压缩到主存储器当中， 并且利用核心的功能，开始测试与驱动各个周边装置，包括储存装置、CPU、网络卡、声卡等等。 那么核心档案在哪里啊？一般来说，他会被放置到 /boot 里面， 并且取名为 /boot/vmlinuz 才对！

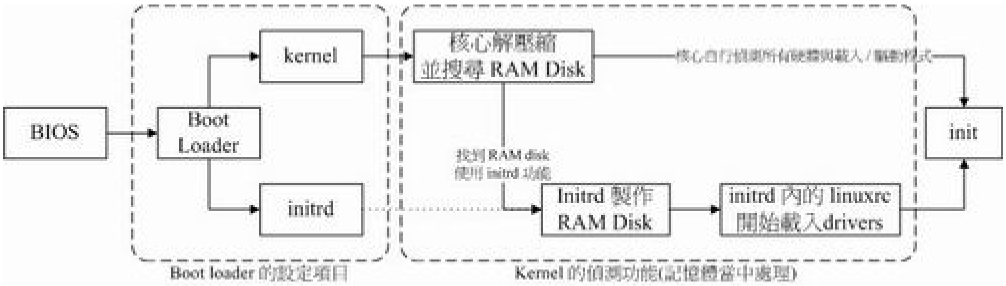
在加载核心的过程当中，我们必须要知道的是，系统只会『挂载根目录』而已，而且是以只读的方式挂载的。 此外，有时为了让某些功能可以用档案的方式来读取，因此，有的系统在开机的时候， 会制作所谓

的虚拟硬盘 (RAM Disk) 来辅助的，那就是 initrd 以及 linuxrc 的功用了。 利用 boot loader 的功能，可以在加载核心的时候，一起加载 initrd 的映象檔 (/boot/initrd-xxxx.img)， Linux 系统会主动的以 initrd (man 4 initrd) 来进行虚拟硬盘的建置， 并且利用 linuxrc (包含在 initrd 的映象档内) 这个程序的功能来进行加载模块的动作。 linuxrc 主要的特性是：

- 必须是 linuxrc 这个档名；
- 必须放置在 initrd 所建立的虚拟磁盘的最顶层目录；
- 必须要可以被核心所执行。

在核心驱动周边硬件的工作完成之后， initrd 所建立的虚拟磁盘就会被移除了！ 不过您要注意的是， initrd 并非必要的，是可有可无的，要看您当初建立该核心的时候， 整个编译的角度与过程。一般来说，各大 Linux distributions 在建立核心时， 都会一起建立出这个 initrd 的映象档，辅助开机的顺利进行。

总之，在这个过程当中， boot loader 可以找到 Linux 的核心档案并且将他加载到主存储器当中， 同时可能可以藉由 initrd 建立起虚拟硬盘 (RAM Disk) 辅助开机的进行， 最后，将读自 BIOS 的主机硬件数据交由 Linux 核心来进行侦测并且加载适当的驱动程序 (driver) ，就让整个主机硬件准备系统的要求了。整个流程有点像这样：



图一、BIOS 与 boot loader 及核心加载流程示意图

在核心完整的加载后，您的主机应该就开始正确的运作了，接下来，就是要开始执行系统的第一支程序： init。

第一支程序 init 及设定文件 /etc/inittab 与 runlevel

在核心加载完毕之后，此时系统应该就已经准备妥当，等待程序的执行了。而整个 Linux 系统当中第一支被执行的程序就是『 /sbin/init 』啰～这也是我们在前一章使用 ps aux |more 时， 看到第一行所显示的程序内容 (PID 为 1 的那行啦) ！ init 这支程序所做的工作相当的多， 他除了利用设定档『 /etc/inittab 』来取得开机的等级 (Run level) 之外，还会经由这个 run level 的设定值来进行不同的开机服务项目的启动。

那么什么是 run level 呢？他有什么功用啊？其实很简单啦， Linux 就是藉由设定 run level 来规定系统使用不同的服务来启动，让 Linux 的使用环境不同。基本上，依据有无网络与有无 X Window 而将 run level 分为六个等级，分别是：

- 0 - halt (系统直接关机)

- 1 - single user mode (单人维护模式，用在系统出问题时的维护)
- 2 - Multi-user, without NFS (类似底下的 runlevel 3，但无 NFS 服务)
- 3 - Full multi-user mode (完整的含有网络功能的纯文字模式)
- 4 - unused (系统保留功能)
- 5 - X11 (与 runlevel 3 类似，但使用 X Window)
- 6 - reboot (重新开机)

由于 run level 0, 4, 6 不是关机、重新开机就是系统保留的，所以：『 您当然不能将预设的 run level 设定为这三个值 』， 否则系统就会不断的自动关机或自动重新开机....

好了，那么我们开机时，到底是如何取得系统的 run level 的？呵呵！当然是 /etc/inittab 所设定的啰！那么 /etc/inittab 到底有什么信息呢？我们先来看看这个档案的内容好了：

```
[root@linux ~]# vi /etc/inittab
# 设定系统开机预设的 run level 设定项目：
id:3:initdefault:

# 开始进行 run level 的服务启动前，使用来侦测与初始化系统环境的设定文件：
si::sysinit:/etc/rc.d/rc.sysinit

# 7 个不同 run level 的，需要启动的服务的 scripts 放置路径：
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# 是否允许按下 [ctrl]+[alt]+[del] 就重新开机的设定项目：
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# 本机端终端机启动的个数：
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# 在 X Window (run level 5) 环境下的启动 script 设定项目：
x:5:once:/etc/X11/prefdm -nodaemon
```

这个档案的语法是这样的：

```
[设定项目]:[run level]:[init 的动作行为]:[指令项目]
```

```
1. 设定项目：
   最多四个字符，代表 init 的主要工作项目，只是一个简单的代表说明。
2. run level：
   该项目在哪些 run level 底下进行的意思。如果是 35 则代表 runlevel 3 与
   5 都会执行。
3. init 的动作项目：
   主要可以进行的动作项目意义有：
   initdefault      ：代表预设的 run level 设定值；
   sysinit：代表系统初始化的动作项目；
   ctrlaltdel       ：代表 [ctrl]+[alt]+[del] 三个按键是否可以重新开机的设定；
   wait             ：代表后面接的指令项目必须要执行完毕才能继续后面的动作；
   respawn：代表后面接的， init 仍会主动的『重新』启动。
   更多的设定项目请参考 man inittab 的说明。
4. 指令项目：
   亦即应该可以进行的指令，通常是一些 script 啰。
```

所以我们可以得到这样的结论：

- 如果不想让使用者利用 [crtl]+[alt]+[del] 来重新启动系统，可以将底下这一行批注掉：ca::ctrlaltdel:/sbin/shutdown -t3 -r now
- 规定开机的预设 run level 是纯文字 (3) 或者是具有图形接口 (X Window, 5) ，可经由『 id:3:initdefault: 』那个数字来决定！ 以鸟哥自己这个档案为例，我是使用纯文字喔！

所以说，你现在会自行修改登入时的预设 run level 设定值了吗？够简单的吧？ 一般来说，我们预设都是 3 或者是 5 来作为预设的 run level 的。但有时后可能需要进入 run level 1， 也就是单人维护模式的环境当中。这个 run level 1 有点像是 Windows 系统当中的『安全模式』啦， 专门用来处理当系统有问题时的操作环境。此外，当系统发现有问题时，举例来说，不正常关机造成 filesystem 的不一致现象时，系统会主动的进入单人维护模式呢！

好了， init 在取得 run level 之后，接下来要干嘛？ 上面 /etc/inittab 档案内容不是有提到 sysinit 吗？嘿嘿！准备初始化系统了吧！

---

init 处理系统初始化流程 (/etc/rc.d/rc.sysinit)

还记得上面提到 /etc/inittab 里头有这一句『 si::sysinit:/etc/rc.d/rc.sysinit 』吧？ 这表示：『我开始加载各项系统服务之前，得先做好整个系统环境，我主要利用 /etc/rc.d/rc.sysinit 这个 shell script 来设定好我的系统环境的。』够清楚了吧？ 所以，我想要知道到底 FC4 开机的过程当中帮我进行了什么动作， 就得要仔细的分析 /etc/rc.d/rc.sysinit 啰。

> Tips:
> 老实说，这个档案的档名在各不同的 distributions 当中都不相同， 例如 SuSE server 9 就使用 /etc/init.d/boot 与 /etc/init.d/rc 来进行的。 所以，你最好还是自行到该档案去察看一下系统的工作喔！ ^_^



/etc/rc.d/rc.sysinit 主要的工作大抵有这几项：

1. 取得网络环境与主机类型：
   首先读取网络设定文件 /etc/sysconfig/network，取得主机名称与预设通讯闸 (gateway) 等网络环境。
2. 测试与挂载内存装置 /proc 及 USB 装置 /sys：
   除挂载内存装置 /proc 之外，还会主动侦测系统上是否具有 usb 的装置， 若有则会主动加载 usb 的驱动程序，并且尝试挂载 usb 的档案系统。
3. 决定是否启动 SELinux ：
   近期以来，很多 distributions 都加入了美国国家安全局发展的 Security Enhance Linux 套件，这个 SELinux 可以更加强化 Linux 操作环境的安全性，不过，由于安全挂帅， 对于新手来说，不是很容易上手。因此，我们才会建议大家先不要启动啊。无论如何， 在这个阶段我们可以分析 SELinux 是否要启动。
4. 接口设备的侦测与 Plug and Play (PnP) 参数的测试：
   根据核心在开机时侦测的结果 (/proc/sys/kernel/modprobe) 开始进行 ide / scsi / 网络 / 音效 等接口设备的侦测，以及利用以加载的核心模块进行 PnP 装置的参数测试。
5. 使用者自订模块的加载
   使用者可以在 /etc/sysconfig/modules/*.modules 加入自订的模块， 则此时会被加载到系统当中喔！
6. 加载核心的相关设定：
   系统会主动去读取 /etc/sysctl.conf 这个档案的设定值，使核心功能成为我们想要的样子。
7. 设定系统时间 (clock)：
8. 设定终端机 (console) 字形：
9. 设定 RAID 与 LVM 等硬盘功能：
10. 以 fsck 检验磁盘档案系统：
11. 进行磁盘配额 quota 的转换 (非必要)：
12. 重新以可读取模式挂载系统磁盘：
13. 启动 quota 功能：
14. 启动系统随机数装置 (产生随机数功能)：
15. 清除开机过程当中的临时文件：
16. 将开机相关信息加载 /var/log/dmesg 档案中。

如此一来，在 /etc/rc.d/rc.sysinit 就已经将基本的系统设定数据都写好了，也将系统的数据设定完整！而如果你想要知道到底开机的过程中发生了什么事情呢？那么就使用 dmesg 就可以知道啰。 另外，基本上，在这个档案当中所进行的很多工作的预设设定档，其实都在 /etc/sysconfig 当中呢！ 所以，请记得将 /etc/sysconfig 内的档案好好的瞧一瞧喔！ ^_^

在这个过程当中，比较值得注意的是自订模块的加载！在 FC4 当中，如果我们想要加载核心模块的话， 可以将整个模块写入到 /etc/sysconfig/modules/*.modules 当中，在该目录下， 只要记得档名最后是以 .modules 结尾即可。 这个过程是非必要的，因为我们目前的预设模块实在已经很够用了，除非是您的主机硬件实在太新了， 非要自己加载新的模块不可，否则，在经过 /etc/rc.d/rc.sysinit 的处理后， 你的主机系统应该是已经跑得很顺畅了啦！就等着你将系统相关的服务与网络服务启动啰！

启动系统服务与相关启动设定档 (/etc/rc.d/rc.n & /etc/sysconfig)

加载核心让整个系统准备接受指令来工作，然后再经过 /etc/rc.d/rc.sysinit 的系统模块与相关硬件信息的初始化后，你的 FC4 系统应该已经顺利工作了。 只是，我们还得要启动系统所需要的各项『服务』啊！这样主机才能提供我们相关的网络或者是主机功能嘛！ 这个时候，依据我们在 /etc/inittab 里面提到的 run level 设定值，就可以来决定启动的服务项目了。 举例来说，使用 run level 3 当然就不需要启动 X Window 的相关服务啰，您说是吧？

那么各个不同的 run level 服务启动的各个 shell script 放在哪？还记得 /etc/inittab 里面提到的：

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

上面提到的就是各不同 run level 放置的目录啦！举例来说， run level 3 的启动目录就是放在 /etc/rc.d/rc3.d 目录当中啰～当然啦，不同的 distributions 这个目录可能会有差异， 所以，您还是得要自行到 /etc/inittab 里面瞧一瞧先！那么在这个目录当中有什么咚咚啊？ 我们先以鸟哥自己的宿舍的 FC4 主机里头的 run level 3 的启动目录瞧一瞧：

```
[root@linux ~]# ls -l /etc/rc.d/rc3.d
lrwxrwxrwx  1 root root 13 Jun 29 01:05 K01yum -> ../init.d/yum
lrwxrwxrwx  1 root root 19 Jun 29 01:05 K02haldaemon -> ../init.d/haldaemon
.....中间省略......
lrwxrwxrwx  1 root root 17 Sep 16 14:09 S01sysstat -> ../init.d/sysstat
lrwxrwxrwx  1 root root 17 Jun 29 01:05 S10network -> ../init.d/network
lrwxrwxrwx  1 root root 16 Jun 29 01:05 S12syslog -> ../init.d/syslog
.....中间省略......
lrwxrwxrwx  1 root root 11 Jun 25 08:27 S99local -> ../rc.local
```

在这个目录下的档案很有趣，全部都是以 S 或者是 K 为开头的档案，而且全部都是连结档， 连结到 /etc/rc.d/init.d 里面的 shell script 呢！而在 /etc/rc.d/init.d 这个目录其实与 /etc/init.d 是一样的，因为这两个目录是连结文件啊！要注意的是，在 /etc/rc.d/init.d/ 底下的 shell scripts 都使用 case.....esac 的语法，而且支持的变量 ($1) 主要有 start 及 stop ， 相关的 shell script 请您回到第三篇去复习。所以，一般来说，如果我们想要启动一些系统服务，例如启动 atd ， 需要使用：

    /etc/rc.d/init.d/atd start (也可以用 /etc/init.d/atd start)

如果是关闭该服务，就是使用：

    /etc/rc.d/init.d/atd stop

了解鸟哥想要表达的东西了吗？是的～如果我想要在 run level 3 的环境下执行某个服务， 当然就得要将该服务写入 /etc/rc.d/rc3.d 里面去，而既然我们的服务已经在 /etc/rc.d/init.d 里面建立好了，自然可以使用连结的方式连结到 /etc/rc.d/init.d/ 内的相关的 shell script 啦。不过，为了解决 start 或 stop 这个变量，因此就有了 S 与 K 开头的档名了。

另外，各不同的服务其实还是互有关系的，举例来说，如果要启动 WWW 服务，总是得要有网络吧？ 所以

啰， /etc/rc.d/init.d/network 就会比较先被启动啦！那么您就会知道在 S 或者是 K 后面接的数字是啥意思了吧？嘿嘿，那就是执行的顺序啦！所以说：

- 在 /etc/rc.d/rc3.d 内的，以 S 为开头的档案，为开机时，需要『启动，start』的服务；
- 在该目录内的 K 为开头的档案，为『关机时需要关闭的服务，stop』的档案连结；
- 在 S 与 K 后面接的数字，代表该档案被执行的顺序。

举例来说，在上表当中， S10network 指向 ../init.d/network ，代表：开机时，执行『 /etc/rc.d/init.d/network start 』的意思，而 S12syslog 则代表开机时执行『 /etc/rc.d/init.d/syslog start 』的意思，且 S10network 要比 S12syslog 还要早执行喔！ 所以啰，看到最后一个被执行的项目是啥？呵呵！没错，就是 S99local ，亦即是： /etc/rc.d/rc.local 这个档案啦！

好了，那么问题来了，我要如何建立 /etc/rc.d/init.d 里面的档案呢？ 很简单啊，看一下 /etc/rc.d/init.d/atd 的内容就知道了，而更多的 services 启动与相关说明，我们会在后续的 认识系统服务 详谈。 而将 /etc/rc.d/init.d/ 连结到 /etc/rc.d/rc3.d 的方法，除了手动建立外， 其实我们都是以 chkconfig 这个程序来进行管理的呢！更多的 chkconfig 请参考认识系统服务那一章。

使用者自订开机启动程序 (/etc/rc.d/rc.local)

在完成 run level 3 的服务启动后，如果我还有其它的动作想要完成时，举例来说， 我还想要寄一封 mail 给某个系统管理账号，通知他，系统刚刚重新开机完毕，那么， 是否应该要制作一个 shell script 放置在 /etc/rc.d/init.d/ 里面，然后再以连结方式连结到 /etc/rc.d/rc3.d/ 里面呢？呵呵！当然不需要！还记得上一小节提到的 /etc/rc.d/rc.local 吧？ 这个档案就可以执行您自己想要执行的系统指令了。像不像早期 DOS 年代的 autoexec.bat 与 config.sys 呢？ ^_^

也就是说，我有任何想要在开机时就进行的工作时，直接将他写入 /etc/rc.d/rc.local ， 那么该工作就会在开机的时候自动被加载喔！而不必等我们登入系统去启动呢！ 是否很方便啊！一般来说，鸟哥就很喜欢把自己制作的 shell script 完整档名写入 /etc/rc.d/rc.local ，如此一来，开机就会将我的 shell script 执行过，真是好棒那！

根据 /etc/inittab 之设定，加载终端机或 X-Window 接口。

在完成了系统所有服务的启动后，接下来 Linux 就会启动终端机或者是 X Window 来等待使用者登入啦！实际参考的项目是 /etc/inittab 内的这一段：

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

```
# Run xdm in runlevel 5
x:5:once:/etc/X11/prefdm -nodaemon
```

这一段代表，在 run level 2, 3, 4, 5 时，都会执行 /sbin/mingetty 这个咚咚， 而且执行六个，这也是为何我们 Linux 会提供『六个纯文字终端机』的设定所在啊！ 因为 mingetty 就是在启动终端机的指令说。

要注意的是那个 respawn 的 init 动作项目，他代表『当后面的指令被终止 (terminal) 时， init 会主动的重新启动该项目。』这也是为何我们登入 tty1 终端机接口后，以 exit 离开后， 系统还是会重新显示等待使用者输入的画面的原因啊！

如果改天您不想要有六个终端机时，可以取消某些终端机接口吗？当然可以啊！ 就将上面表格当中的某些项目批注掉即可！例如不想要 tty5 与 tty6 ，就将那两行批注， 则下次重新开机后，您的 Linux 就只剩下『 F1 ~ F4 』有效而已，这样说，可以了解吧！！ ^_^

至于如果我们使用的是 run level 5 呢？那么除了这六个终端机之外， init 还会执行 /etc/X11/prefdm -nodaemon 那个指令喔！该指令我们会在 X Window 章节再来详谈！ 他主要的功能就是在启动 X Window 啦！

其它开机相关事项：

- 关于模块： /etc/modprobe.conf

还记得我们在 /etc/rc.d/rc.sysinit 当中谈到的加载使用者自订模块的地方吗？嘿嘿！就是在 /etc/sysconfig/modules/ 目录下啊！ 不过，虽然核心提供的预设模块已经很足够我们使用了，但是， 某些条件下我们还是得对模块进行一些参数的规划，此时，就得要使用到 /etc/modprobe.conf 啰！ 举例来说，鸟哥的 FC4 主机的 modprobe.conf 有点像这样：

```
[root@linux ~]# vi /etc/modprobe.conf
alias eth0 8139too
alias snd-card-0 snd-via82xx
options snd-card-0 index=0
options snd-via82xx index=0
alias usb-controller uhci-hcd
```

意思是说：『我的 eth0 这个玩意儿，代表的是使用 8139too 这个核心模块， 至于 snd-card-0 则使用 snd-via82xx 那个模块。此外， snd-card-0 这个模块在使用时， 还使用 index=0 这个参数。』这玩意真的是挺常用的～不过，这个档案通常在安装的时候， 安装程序就会主动的建立这个档案啰～除非您对系统提供的驱动程序模块不满意～～ 才会主动的修改这个模块加载的相关档案啦～(早期 2.4.xx 核心版本时，使用的是 /etc/modules.conf 喔！) 更多的相关说明，请 man modprobe.conf 喔！

- /etc/sysconfig/*

不说您也知道，整个开机的过程当中，老是读取的一些服务的相关设定文件都是记录在 /etc/sysconfig 目录下的！那么该目录底下有些啥玩意儿？我们先来瞧一瞧！

```
[root@linux ~]# ls -l /etc/sysconfig
```

```
-rw-r--r--  1 root root  194 Jun 25 08:53 authconfig
-rw-r--r--  1 root root  726 Apr 25 23:54 autofs
-rw-r--r--  1 root root   39 Jun 25 16:55 clock
drwxr-xr-x  2 root root 4096 May 26 00:52 console
-rw-r--r--  1 root root  512 Jul 12 06:21 crond
-rw-r--r--  1 root root   14 Jun 25 08:53 desktop
-rw-r--r--  1 root root   31 Aug 23 03:13 diskdump
-rw-r--r--  1 root root   17 Jun 25 16:56 firstboot
-rw-r--r--  1 root root   25 Jun 25 08:53 grub
-rw-r--r--  1 root root 1592 Mar  2  2005 harddisks
-rw-r--r--  1 root root  112 Jun 25 19:53 i18n
-rw-r--r--  1 root root  991 Nov  2  2004 init
-rw-------  1 root root 1376 Mar 19  2005 iptables-config
-rw-r--r--  1 root root  180 Jun 25 08:53 kernel
-rw-r--r--  1 root root   32 Jun 25 08:53 keyboard
-rw-r--r--  1 root root  168 May 20 03:54 kudzu
drwxr-xr-x  2 root root 4096 May 26 00:52 modules
-rw-r--r--  1 root root  115 Jun 25 08:53 mouse
-rw-r--r--  1 root root   43 Jun 25 08:53 network
drwxr-xr-x  4 root root 4096 Jun 25 08:27 networking
drwxr-xr-x  2 root root 4096 Jun 25 21:53 network-scripts
-rw-r--r--  1 root root  454 May 20 00:07 syslog
-rw-r--r--  1 root root   66 Mar  7  2005 sysstat
-rw-r--r--  1 root root  376 Mar  9  2005 xinetd
```

为了节省篇幅，上表当中我已经省略掉某些档案了，仅列出较重要的几个！ 需要注意的是：

- authconfig：
  这个档案主要在规范使用者的身份认证，包括加密与否、加密的机制等；
- clock：
  此档案在设定 Linux 主机的时区，可以使用格林威治时间(GMT) ，也可以使用台湾的本地时间 (local)。基本上，在 clock 档案内的设定项目『 ZONE 』所参考的时区位于 /usr/share/zoneinfo 目录下的相对路径中。 而且要修改时区的话，还得将 /usr/share/zoneinfo/Asia/Taipei 这个档案复制成为 /etc/localtime 才行！
- desktop：
  这个与预设的 X Window 的窗口管理员 (Window Manager) 有关。 在 FC4 里头预设是以 KDE 为主要的 WM，您也可以自行在这个档案内修订喔！
- i18n：
  i18n 在设定一些语系的使用方面，例如最麻烦的文字接口下的日期显示问题！ 如果您是以中文安装的，那么预设语系会被选择 big5 ，所以在纯文字接口之下， 你的档案日期显示就会呈现乱码！这个时候就需要更改一下这里啦！更动这个 i18n 的档案，将里面的 LC_TIME 改成 en 即可！
- keyboard & mouse：
  keyboard 与 mouse 就是在设定键盘与鼠标的形式；
- network：
  network 可以设定主机名称，以及 GATEWAY 这两个重要信息呢！

- network-scripts/:
  至于 network-scripts 里面的档案，则是主要用在设定网络卡～ 这部份我们在服务器架设篇才再次提到！

总而言之一句话，这个目录下的档案很重要的啦！开机过程里面常常会读取到的！

Run level 之变换：

在我们完成上面的所有信息后，其实整个 Linux 主机就已经在等待我们使用者的登入啦！ 但是，相信您应该还是会有一点疑问的地方，那就是：『我该如何切换 run level 呢？』会不会很难啊？不会啦！很简单～ 但是依据执行的时间而有不同的方式啊！

事实上，与 run level 有关的启动其实是在 /etc/rc.d/rc.sysinit 执行完毕之后。也就是说，其实 run level 的不同仅是 /etc/rc.d/rc[0-6].d 里面启动的服务不同而已。不过，依据开机是否自动进入不同 run level 的设定，我们可以说：

1. 要每次开机都执行某个预设的 run level ，则需要修改 /etc/inittab 内的设定项目， 亦即是『 id:3:initdefault: 』里头的数字啊；
2. 如果仅只是暂时变更系统的 run level 时，则使用 init [0-6] 来进行 run level 的变更。 但下次重新开机时，依旧会是以 /etc/inittab 的设定为准。

假设原本我们是以 run level 5 登入系统的，但是因为某些因素，想要切换成为 run level 3 时， 该怎么办呢？很简单啊，利用 init 3 即可切换。但是 init 3 这个动作到底做了什么呢？ 我们不是说了吗？事实上，不同的 run level 只是加载的服务不同罢了， 亦即是 /etc/rc.d/rc5.d/ 还有 /etc/rc.d/rc3.d 内的 Sxxname 与 Kxxname 有差异而已。 所以说，当执行 init 3 时，系统会：

- 先比对 /etc/rc.d/rc3.d/ 及 /etc/rc.d/rc5.d 内的 K 与 S 开头的档案；
- 关闭 /etc/rc.d/rc5.d/ 内不存在于 /etc/rc.d/rc3.d/ 中的服务；
- 启动 /etc/rc.d/rc3.d/ 内不存在于 /etc/rc.d/rc5.d/ 中的服务。

也就是说，两个 run level 都存在的服务就不会被关闭啦！如此一来，就很容易切换 run level 了， 而且还不需要重新开机呢！真方便。那我怎么知道目前的 run level 是多少呢？ 直接在 bash 当中输入 runlevel 即可啊！

```
[root@linux ~]# runlevel
N 3
```

够简单的吧！ ^_^

核心与核心模块：

谈完了整个开机的流程，您应该会知道，在整个开机的过程当中，是否能够成功的驱动我们主机的硬件配备， 是核心 (kernel) 的工作！而核心一般都是压缩档，因此在使用核心之前，就得要将他解压缩后， 才能加载主存储器当中。

另外，为了应付日新月异的硬件，目前的核心都是具有『可读取模块化驱动程序』的功能， 亦即是所谓的

『 modules (模块化)』的功能啦！所谓的模块化可以将他想成是一个『外挂程序』， 该外挂程序可能由硬件开发厂商提供，也有可能我们的核心本来就支持～不过，较新的硬件， 通常都需要硬件开发商提供驱动程序模块啦！

那么核心与核心模块放在哪？

- 核心： /boot/vmlinuz 或 /boot/vmlinuz-version；
- 核心解压缩所需 RAM Disk： /boot/initrd (/boot/initrd-version)；
- 核心模块： /lib/modules/version/kernel 或 /lib/modules/`uname -r`/kernel；
- 核心原始码： /usr/src/linux (要安装才会有！否则预设不安装的！)

如果该核心被顺利的加载系统当中了，那么就会有几个信息纪录下来：

- 核心版本： /proc/version
- 系统核心功能： /proc/sys/kernel

问题来啦，如果我有个新的硬件，偏偏我的操作系统不支持，该怎么办？很简单啊！

- 重新编译核心，并加入最新的硬件驱动程序原始码；
- 将该硬件的驱动程序编译成为模块，在开机时加载该模块

上面第一点还很好理解，反正就是重新编译核心就是了。不过，核心编译很不容易啊！ 我们会在后续章节约略介绍核心编译的整个程序。比较有趣的则是将该硬件的驱动程序编译成为模块啦！ 关于编译的方法，可以参考后续的 原始码与 tarball 那一章的介绍。 我们这个章节仅是说明一下，如果想要加载一个已经存在的模块时，该如何是好？

核心模块与相依性：

既然要处理核心模块，自然就得要了解了解我们核心提供的模块之间的相关性啦！ 基本上，核心的放置处是在 /lib/modules/`uname -r`/kernel 当中，里面主要还分成几个目录：

```
arch       ：与硬件平台有关的项目，例如 CPU 的等级等等；
crypto     ：核心所支持的加密的技术，例如 md5 或者是 des 等等；
drivers    ：一些硬件的驱动程序，例如显示卡、网络卡、PCI 相关硬件等等；
fs         ：核心所支持的 filesystems ，例如 vfat, reiserfs, nfs 等等；
lib        ：一些函式库；
net        ：与网络有关的各项协议数据，还有防火墙模块 (net/ipv4/netfilter/*) 等等；
sound      ：与音效有关的各项模块；
```

如果要我们一个一个的去检查这些模块的主要信息，然后定义出他们的相依性， 我们可能会疯掉吧！所以说，我们的 Linux 当然会提供一些模块相依性的解决方案啰～ 对啦！那就是检查 /lib/modules/`uname -r`/modules.dep 这个档案啦！他记录了在核心支持的模块的各项相依性。

那么这个档案如何建立呢？挺简单！利用 depmod 这个指令就可以达到建立该档案的需求了！

```
[root@linux ~]# depmod [-Ane]
参数：
-A  ：不加任何参数时， depmod 会主动的去分析目前核心的模块，并且重新写入
      /lib/modules/`uname -r`/modules.dep 当中。若加入 -A 参数时，则 depmod
      会去搜寻比 modules.dep 还要新的模块，如果真找到新模块，才会更新。
-n  ：不写入 modules.dep ，而是将结果输出到屏幕上(standard out)；
-e  ：显示出目前已加载的不可执行的模块名称
范例：

范例一：若我已经做好一个网络卡驱动程序，假设文件名为 a.ko，该如何更新核心相依性？
[root@linux ~]# cp /full/path/a.ko /lib/modules/`uname -r`/kernel/drivers/net
[root@linux ~]# depmod
```

难就难在将那个新的驱动程序模块编译出来，如果编译出来之后， 依据核心模块放置的目录去放置好，然后输入 depmod 后，去更新好 modules.dep ， 如此一来，核心就能够认识该模块啰！够简单吧！ ^_^ (关于核心模块的编译，请参考 核心编译 一文！)

核心模块的观察： lsmod, modinfo

那你到底晓不晓得目前核心加载了多少的模块呢？粉简单啦！利用 lsmod 即可！

```
[root@linux ~]# lsmod
Module                  Size  Used by
loop                   18121  0
ipt_state               1857  2
ipt_MASQUERADE          3265  2
iptable_filter          2881  1
ip_nat_irc              2753  0
ip_conntrack_irc       72401  1 ip_nat_irc
ip_nat_ftp              3393  0
ip_conntrack_ftp       73297  1 ip_nat_ftp
....中间省略.....
8139too                30017  0
mii                     5441  1 8139too
floppy                 65141  0
ext3                  132681  4
jbd                    86233  1 ext3
```

使用 lsmod 之后，系统会显示出目前已经存在于核心当中的模块，显示的内容包括有：

- 模块名称(Module)；
- 模块的大小(size)；
- 此模块是否被其它模块所使用 (Used by)。

举例来说，上面的表格当中，我的 ip_conntrack_ftp 模块其实还被 ip_nat_ftp 模块所使用呢！ 也就是说，这两个模块之间应该是有相关性的！所以啰，如果我加载 ip_nat_ftp 势必还得要加载 ip_conntrack_ftp 才行～而这个相依性就是被纪录在上个小节提到的 modules.dep 档案内啰！ ^_^

那么除了显示出目前的模块外，我还可以查阅每个模块的信息吗？当然可以啦！就用 modinfo 即可：

```
[root@linux ~]# modinfo [-adln] [module_name|filename]
参数：
-a  ：仅列出作者名称；
-d  ：仅列出该 modules 的说明 (description)；
-l  ：仅列出授权 (license)；
-n  ：仅列出该模块的详细路径。
范例：

范例一：由上个表格当中，请列出 8139too 这个模块的相关信息：
[root@linux ~]# modinfo 8139too
filename:       /lib/modules/2.6.12-1.1398_FC4/kernel/drivers/net/8139too.ko
author:         Jeff Garzik
description:    RealTek RTL-8139 Fast Ethernet driver
license:        GPL
version:        0.9.27
parmtype:       multicast_filter_limit:int
parmtype:       media:array of int
parmtype:       full_duplex:array of int
parmtype:       debug:int
parm:           debug:8139too bitmapped message enable number
parm:           media:8139too: Bits 4+9: force full duplex, bit 5: 100Mbps
parm:           full_duplex:8139too: Force full duplex for board(s) (1)
vermagic:       2.6.12-1.1398_FC4 686 REGPARM 4KSTACKS gcc-4.0
depends:        mii
alias:          pci:v000010ECd00008139sv*sd*bc*sc*i*

范例二：我有一个模块名称为 a.ko ，请问该模块的信息为？
[root@linux ~]# modinfo a.ko
.......省略......
```

事实上，这个 modinfo 除了可以『查阅在核心内的模块』之外，还可以检查『某个模块档案』， 因此，如果你想要知道某个档案代表的意义为何，利用 modinfo 加上完整檔名吧！ 看看就晓得是啥玩意儿啰！ ^_^

核心模块的加载与移除： insmod, modprobe, rmmod

好了，如果我想要自行手动加载模块，又该如何是好？ 有很多方法啦，最简单而且建议的，是使用 modprobe 这个指令来加载模块， 这是因为 modprobe 会主动的去搜寻 modules.dep 的内容，先克服了模块的相依

性后， 才决定需要加载的模块有哪些，很方便。至于 insmod 则完全由使用者自行加载一个完整文件名的模块， 并不会主动的分析模块相依性啊！

```
[root@linux ~]# insmod [/full/path/module_name] [parameters]

范例一：请尝试载入 /lib/modules/`uname -r`/kernel/fs/smbfs/smbfs.ko
[root@linux ~]# insmod /lib/modules/`uname -r`/kernel/fs/smbfs/smbfs.ko
[root@linux ~]# lsmod | grep smbfs
smbfs                  67897  0
```

对吧！他立刻就将该模块加载啰～这个需要加入完整档名啦！那如何移除这个模块呢？

```
[root@linux ~]# rmmod [-fw] module_name
参数：
-f  ：强制将该模块移除掉，不论是否正被使用；
-w  ：若该模块正被使用，则 rmmod 会等待该模块被使用完毕后，才移除他！
范例：

范例一：将刚刚加载的 smbfs 模块移除！
[root@linux ~]# rmmod smbfs
```

帅吧！移除掉了。不过，如前所述的， insmod 实在不怎么人性化，近年来， 我们都建议直接使用 modprobe 来处理模块加载的问题，这个指令的用法是：

```
[root@linux ~]# modprobe [-lcf] module_name
参数：
-c  ：列出目前系统所有的模块！(更详细的代号对应表)
-l  ：列出目前在 /lib/modules/`uname -r`/kernel 当中的所有模块完整文件名；
-f  ：强制加载该模块；
-r  ：类似 rmmod ，就是移除某个模块啰～
范例：

范例一：加载 smbfs 模块
[root@linux ~]# modprobe smbfs
# 很方便吧！不需要知道完整的模块文件名，这是因为该完整文件名已经记录到
# /lib/modules/`uname -r`/modules.dep 当中的缘故啊！如果要移除的话：
[root@linux ~]# modprobe -r smbfs
```

使用 modprobe 真的是要比 insmod 方便很多！因为他是直接去搜寻 modules.dep 的纪录， 所以啰，当然可以克服模块的相依性问题，而且还不需要知道该模块的详细路径呢！ 好方便！ ^_^

核心模块的额外参数设定： /etc/modprobe.conf

这个档案我们之前已经谈过了，这里只是再强调一下而已，如果您想要修改某些模块的额外参数设定， 就在这个档案内设定吧！我们假设一个案例好了，假设我的网络卡 eth0 是使用 ne ， 但是 eth1 同样也使用 ne ，为了避免同一个模块会导致网络卡的错乱， 因此，我可以先找到 eth0 与 eth1 的 I/O 与 IRQ ，假设：

- eth0 ： I/O (0x300) 且 IRQ=5
- eth1 ： I/O (0x320) 且 IRQ=7

则：

```
[root@linux ~]# vi /etc/modprobe.conf
alias eth0 ne
alias eth1 ne
options eth0 io=0x300 irq=5
options eth1 io=0x320 irq=7
```

嘿嘿！如此一来，我的 Linux 就不会捉错网络卡的对应啰！因为被我强制指定某个 I/O 咯嘛！ ^_^

Boot Loader: Grub

在看完了前面的整个开机流程，以及核心模块的整理之后，你应该会发现到一件事情， 那就是『 boot loader 是载入核心的重要工具』啊！没有 boot loader 的话，那么 kernel 根本就没有办法被系统加载的呢！所以，底下我们会先谈一谈 boot loader 的功能， 然后再讲一讲现阶段 Linux 里头最主流的 grub 这个 boot loader 吧！

---

boot loader 的功能与意义：

我们在第一小节的地方，曾经讲过，在 BIOS 读完信息后，接下来就是会到第一个开机装置的 MBR 去读取 boot loader 了，这个 boot loader 可以具有选单功能，而且『还能辨识硬盘的 filesystem ， 并且指向核心档案，以将他读入主存储器当中』呢！所以啰，特点是： 我们系统能够使用的 boot loader 必须要能够认识我们系统的 filesystem 才行。 目前台湾常见的有 grub, lilo 以及 spfdisk 这几个 loader 啦！

但是我们都知道， MBR 是整个硬盘的第一个 sector，充其量整个大小不可能超过 512 bytes 的，那么，我们的 loader 功能这么强，不可能只占不到 512 bytes 的容量吧？ 而且某些情况下，设定档还会占用掉不少的容量呢！怎么办？

为了解决这个问题，我们将 boot loader 分成两个阶段来执行 (stage)：

- Stage 1：第一阶段为 boot loader 的主程序，这个主程序必须要被安装在开机区， 亦即是 MBR 或者是 Super block (first sector)。但如前所述，因为 MBR 实在太小了， 所以，这个 stage 1 通常仅安装 boot loader 的最小主程序， 并没有安装 loader 的相关设定档；
- Stage 2：第二阶段为加载 boot loader 的所有设定档与相关的环境参数档案。 一般来说，设定档都在 /boot 底下。

另外，不知道你有没有觉得很奇怪，既然我们可以将 boot loader 安装在 super block (可以想成是每个 partition 的第一个扇区 "first sector" ，更多相关信息，请参考 磁盘档案系统 那个章节。)，然后开机时，主要的 loader 又是加载自 MBR ，那么 Super block 的 boot loader 什么时候会被使用到啊？

果然是好问题～如果这个地方搞懂了，你的主机多重开机就可以搞定啰～不过，最难懂得却也是这个地方。其实针对开机的项目， boot loader 可以做到：

- boot loader 可以直接指定并取用 kernel 档案，来加载到主存储器当中；
- 也可以将 loader 的控制权移交给下一个 loader ！

换句话说， boot loader 除了可以直接指定核心档案来开机之外，也可以指定某个 super block 当中的 boot loader 接管开机的核心加载流程啊！我们来假设几个条件好了。 假设我在 MBR 安装了 grub 这个同时认识 Windows 与 Linux 的档案系统的 boot loader ， 同时假设我的 /dev/hda2 当中的 super block 也安装了 Linux 的 grub ， 且 /dev/hda1 的 super block 则是安装 Windows 的 boot loader 。此外，我的 Linux 的核心档案放置在 /dev/hda2 里面的 /boot/vmlinuz ，那么我的 MBR 的 grub 至少可以做到这样：

- 直接指定核心 (在 /dev/hda2 的 /boot/vmlinux) 来进行开机；
- 将控制权交给 /dev/hda2 super block 当中的 grub 进行管理；
- 将控制权交给 /dev/hda1 super block 当中的 Windows 的 loader 来管理。

这样说，瞭了吗？而值得注意的是，我们的 Linux 可以选择将 boot loader 安装在 MBR 或者是 super block 当中，但是 Windows 系统则几乎预设强制会同时安装在 MBR 与 Super block 当中， 这也是为什么『我们说要安装多重操作系统时，最好先安装 Windows 再安装 Linux ，因为若先安装 Linux ，则后续安装 Windows 时，会强制将 MBR 的 boot loader 覆盖掉，如此一来，我们将无法以 windows 的 boot loader 进入 Linux 了。』

但如果我真的是忘记了，先安装 Linux 后才安装 Windows 呢？怎么办？ 没关系啊！只要你安装类似 spfdisk 的软件在 MBR 里面，因为他同时认识 Linux 与 Windows ， 所以就可以用他来进入 Linux 啦！或者使用类似 KNOPPIX 的 Live CD 以光盘开机进入 Linux 之后， 再以 chroot 软件切换根目录 (/) ，然后重新安装 grub 等 boot loader ， 同样也可以重新让两个操作系统存在啦！总之， 只要你知道 MBR / Super block / boot loader 之间的相关性，怎么切换都可能啊！ ^_^

---

grub 的设定档 /boot/grub/menu.lst 与安装型态

grub 是较新的 boot loader ，他的优点很多，包括：

- 认识与支持较多的 filesystem ，并且可以使用 grub 的主程序直接在 filesystem 当中搜寻核心；
- 开机的时候，可以『自行编辑与修改开机设定项目』，类似 bash 的指令模式；
- 可以动态搜寻设定文件，而不需要在修改设定档后重新安装 grub 。亦即是我们只要修改完 /boot/grub/menu.lst 里头的设定后，下次开机就生效了！

上面第三点其实就是 Stage 1, Stage 2 分别安装在 MBR 与 filesystem 当中的原因啦！ 好了，接下来，让我们好好了解一下 grub 的设定档： /boot/grub/menu.lst 这玩意儿吧！ 要注意喔，那个 lst 是 LST 的小写，不要搞错啰！

---

* 与硬盘的关系：

既然 grub 主程序是安装在 MBR (super block) 当中，并且动态去搜寻设定文件的信息， 所以啰，他必须要认识硬盘才行啊！那么 grub 到底是如何认识硬盘的呢？ 嘿嘿！ grub 对硬盘的代号设定与传统的 Linux 磁盘代号可完全是不同的！ 他的代号有点像：

(hd0,0)

够神了吧？跟 /dev/hda1 风马牛不相干～怎么办啊？其实只要注意几个东西即可， 那就是：

* 硬盘代号以小括号 ( ) 包起来；
* 硬盘以 hd 表示，后面会接一组数字；
* 以『搜寻顺序』做为硬盘的编号，而不是依照硬盘排线的排序！(这个重要！)
* 第一个搜寻到的硬盘为 0 号，第二个为 1 号，以此类推；
* 每颗硬盘的第一个 partition 代号为 0 ，依序类推。

所以说，第一颗『搜寻到的硬盘』代号为：『(hd0)』，而该颗硬盘的第一号 partition 为 『(hd0,0)』这样说，容易了解了吧！？在传统的主机板上面， 通常第一颗硬盘就会是 /dev/hda，所以常常我们可能会误会 /dev/hda 就是 (hd0) ， 其实不是喔！要看您 BIOS 的设定值才行！ 有的主机板 BIOS 可以调整开机的硬盘搜寻顺序，那么就要注意了，因为 grub 的硬盘代号可能会跟着改变呐！留意留意！ 所以说，整个硬盘代号为：

硬盘搜寻顺序	在 Grub 当中的代号
第一颗	(hd0) (hd0,0) (hd0,1) (hd0,4)....
第二颗	(hd1) (hd1,0) (hd1,1) (hd1,4)....
第三颗	(hd2) (hd2,0) (hd2,1) (hd2,4)....

这样应该比较好看出来了吧？第一颗硬盘的 MBR 安装处的硬盘代号就是『(hd0)』， 而第一颗硬盘的第一个 partition 的 Super block 代号就是『(hd0,0)』第一颗硬盘的第一个 logical partition 的 super block 代号为『(hd0,4)』瞭了吧！

---

* /boot/grub/menu.lst 设定档：

了解了 grub 当中最麻烦的硬盘代号后，接下来，我们就可以瞧一瞧设定档的内容了。先看一下鸟哥的 FC4 内的 /boot/grub/menu.lst 好了：

```
[root@linux ~]# vi /boot/grub/menu.lst
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title Fedora Core (2.6.12-1.1456_FC4)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.12-1.1456_FC4 ro root=/dev/hda1 quiet vga=787
        initrd /boot/initrd-2.6.12-1.1456_FC4.img
title Fedora Core (2.6.11-1.1369_FC4)
```

```
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.11-1.1369_FC4 ro root=/dev/hda1 quiet vga=787
        initrd /boot/initrd-2.6.11-1.1369_FC4.img
```

在 title 以前的前四行，都是属于 grub 的整体设定，包括预设的等待时间与预设的开机项目， 还有显示的画面特性等等的项目。至于 title 后面才是指定开机的核心档案或者是 boot loader 控制权。 在整体设定方面的项目主要常见的有：

- default=0
  这个必须要与 title 作为对照。以上表为例，我们不是有两个 title 吗？按照前后顺序来排列，第一个 title 代表的是 0 ，第二个 title 代表的是 1 ，以此类推～ 这个 default 说的是，如果开机过程当中，您并没有选择其它的项目， 那么就会用默认值 (第 1 个 title) 来开机啦！
- timeout=5
  这个是开机时，不是会显示选单吗？如果你在几秒内(单位就是秒！)没有按下任何按键， 那就会用 default 那个设定值来进行开机！
- splashimage=(hd0,0)/boot/grub/splash.xpm.gz
  这个 splashimage 是在选单上面显示的一些图片或者是相关的影像数据啦！ 这个设定有个地方比较有趣！因为在开机的过程当中并没有硬盘， 所以我们必须要明确的指出某个档案在那个 partition 内的那个目录； 因此，上面的设定说的是：在 (hd0,0) 那个 partition 内的 /boot/grub/splash.xpm.gz 该档案为开机时显示的画面啦！更多 splash 可以参考：http://ruslug.rutgers.edu/~mcgrof/grub-images/
- hiddenmenu
  这个说的是，开机时，是否要显示选单？目前 FC4 预设是不要显示选单， 如果您想要显示选单，那就将这个设定值批注掉！

整体设定的地方大概是这样，而底下那个 title 则是显示开机的设定项目。 如同前一小节提到的，开机时，可以选择(1)直接指定核心档案开机或 (2)将 boot loader 控制权转移到下个 loader (此过程称为 chain-loader)。每个 title 后面接的是 『该开机项目名称的显示』，亦即是在选单出现时，选单上面的名称而已。 那么这两种方式的设定有啥不同呢？

- 1. 直接指定核心开机

既然要指定核心开机，所以当然要找到核心档案啦！此外，有可能还需要用到 initrd 的 RAM Disk 设定档案 (通常是放置在 /boot 底下啊！)。但是如前说的， 尚未开机完成，所以我们必须要以 grub 的硬盘认识方式找出完整的 kernel 与 initrd 檔名才行。 因此，我们可能需要有底下的方式来设定才行！

```
1. 先指定核心档案放置的 partition，再读取档案 (目录树)，
   最后才加入档案的实际文件名与路径 (kernel 与 initrd)；
   假设仅有一颗硬盘，且仅划分出 /dev/hda1 (亦即根目录为 /dev/hda1)而已：
root    (hd0,0)          <==代表核心档案放在那个 partition 当中？
kernel /boot/vmlinuz ro root=/dev/hda1 vga=771
initrd /boot/initrd
# root ：代表的是『核心档案放置的那个 partition 而不是根目录』喔！不要搞错了！
# kernel ：至于 kernel 后面接的则是核心的档名，而在档名后面接的则是核心的参数，
#          在 kernel 后面接的 root 才是『根目录所在的 partition 』，
#          另外，核心还可以外加很多的参数喔，例如 vga 即是一个分辨率参数！
```

```
# initrd ：就是前面提到的 initrd 制作出 RAM Disk 的档案档名啦！

2. 直接指定 partition 与档名，不需要外接 root ！
kernel (hd0,0)/boot/vmlinuz ro root=/dev/hda1 vga=771
initrd (hd0,0)/boot/initrd
```

注意到：kernel 后面其实只要接『核心档案文件名』与『根目录 (/) 的所在磁盘代号 (用一般 Linux 磁盘代号) 就可以了。老实说，以第二个方式来书写你的 title 的内容会比较好一点～ 不会造成两个 root 是啥意思的紊乱！上面的案例还很好理解，如果是底下的案例呢？ 思考看看：

例题：
我的 Linux 主机仅有一颗硬盘，但为了制作多重开机，所以我将 /boot 独立出来成为一个 partition， partition 的对应是『 /boot → /dev/hda2 』 『 / → /dev/hda1 』，而且我仅有 kernel file， 档名为 /boot/vmlinuz-2.6.11-1.1369_FC4 请问 grub 当中的 title 要如何写？
答：

只要列出 kernel 的檔名即可！因为我将 /boot 独立成为 /dev/hda2 ，因此，整个核心档案档名应该是：

```
/boot/vmlinuz-2.6.11-1.369FC4 -->
(/dev/hda2)/vmlinuz-2.6.11-1.1369_FC4 -->
(hd0,1)/vmlinuz-2.6.11-1.1369_FC4
```

因为 /boot 是一个完整的 partition 嘛！所以说，整个核心档案的写法，可以这样做：

```
title FC4 default
    kernel (hd0,1)/vmlinuz-2.6.11-1.1369_FC4 ro root=/dev/hda1
```

因为 vmlinuz-2.6.11-1.1369_FC4 这个档案其实是在 /boot 所在的 partition 上， 而 /boot 是 (hd0,1) ，因此，整个档名就成为 (hd0,1)/vmlinuz-2.6.11-1.1369_FC4 了！ 只要你能够了解这个档名的来源，那么 grub 对你而言，已经没有什么大问题了！ ^_^

- 2. 利用 chain loader 的方式：

所谓的 chain loader 仅是在将控制权交给下一个 boot loader 而已， 所以 grub 并不需要认识与找出 kernel file ，『 他只是将 boot 的控制权交给下一个 super block 或者是 MBR 内的 boot loader 而已 』 所以通常他也不需要去查验下一个 boot loader 的开机扇区啊！ 一般来说， chain loader 的设定只要两个就够了，一个是指定开机区的 root partition，另一个则是设定 chainloader 在那个扇区上！所以说，假设我的 Windows 扇区在 /dev/hda1 ，且我又只有一颗硬盘， 那么要 grub 将控制权交给 windows 的 loader 只要这样就够了：

```
[root@linux ~]# vi /boot/grub/menu.lst
....前略....
title Windows partition
        root (hd0,0)
```

```
          chainloader +1
```

那个 root 代表的就是 Windows 的 C 槽啦!而 chainloader 则是加载 boot loader 的定义值， 那个 +1 代表的是『第一个 sector 』也可以说成 Super block 啊！这样说，理解吗？！ 但其实我们的 grub 功能是很强大的!他还可以隐藏某些 partition 呢！让您的 Windows 不会去读取 Linux 的 partition 啊!举例来说，以上面的例子在延伸， 假设我的 /dev/hda5 是 Linux 的磁盘系统，我想将他隐藏，并且把原先隐藏的 /dev/hda2 开启，并且不去检查 /dev/hda1 的开机区，所以，会变成：

```
[root@linux ~]# vi /boot/grub/menu.lst
....前略....
title Windows partition
          unhide (hd0,1)
          hide (hd0,4)
          rootnoverify (hd0,0)
          chainloader +1
          makeactive
```

最后那个 makeactive 是让开机区的 boot 项目 (记得用 fdisk -l 的显示结果吗？ ^_^) 具有 active 的标志而已啦！有没有加都可以！很简单吧！

这样一来，您对于 grub 的硬盘以及 menu.lst 的设定应该有一定程度的认识了吧？好～ 接下来，让我们实际的依据您的环境来安装啰～ 在下一小节，我们会以鸟哥自己宿舍的计算机来做解释呢！ ^_^

测试与安装 grub

如果你的 Linux 主机本来就是 grub 的话，那么你就不需要重新安装 grub 了， 因为 grub 本来就会主动去读取设定档啊！您说是吧！但如果你的 Linux 原来使用的并非 grub ， 那么就需要来安装啦！如何安装呢？首先，你必须要使用 grub-install 将一些必要的档案复制到 /boot/grub 里面去，你应该这样做的：

```
[root@linux ~]# grub-install [--root-directory=DIR] INSTALL_DEVICE
参数：
--root-directory=DIR 那个 DIR 为实际的目录，使用 grub-install 预设会将
                         grub 所有的档案都复制到 /boot/grub/* 当中，但如果
                         想要复制到其它目录与装置去，就得要用这个参数。
INSTALL_DEVICE 安装的装置代号啦！
范例：

范例一：将 grub 安装在目前系统的 / 底下，我的系统为 /dev/hda：
[root@linux ~]# grub-install /dev/hda
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

# this device map was generated by anaconda
(fd0)     /dev/fd0
```

```
(hd0)     /dev/hda
# 如果去查阅一下 /boot/grub 的内容，会发现所有的档案都更新了，
# 没错啊！因为我们重新安装了嘛！

范例二：我的 /dev/hdb 挂载到 /disk2 下，如何安装 grub 到 /dev/hdb ?
[root@linux ~]# grub-install --root-directory=/disk2 /dev/hdb
Probing devices to guess BIOS drives. This may take a long time.
Installation finished. No error reported.
This is the contents of the device map /disk2/boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(fd0)   /dev/fd0
(hd0)   /dev/hda
(hd1)   /dev/hdb
[root@linux ~]# ll /disk2/boot/grub/
-rw-r--r--  1 root root     45 Sep 27 22:10 device.map
-rw-r--r--  1 root root   7476 Sep 27 22:10 e2fs_stage1_5
-rw-r--r--  1 root root   7300 Sep 27 22:10 fat_stage1_5
-rw-r--r--  1 root root   6612 Sep 27 22:10 ffs_stage1_5
-rw-r--r--  1 root root   6612 Sep 27 22:10 iso9660_stage1_5
-rw-r--r--  1 root root   8096 Sep 27 22:10 jfs_stage1_5
-rw-r--r--  1 root root   6772 Sep 27 22:10 minix_stage1_5
-rw-r--r--  1 root root   8980 Sep 27 22:10 reiserfs_stage1_5
-rw-r--r--  1 root root    512 Sep 27 22:10 stage1
-rw-r--r--  1 root root 101704 Sep 27 22:10 stage2
-rw-r--r--  1 root root   6952 Sep 27 22:10 ufs2_stage1_5
-rw-r--r--  1 root root   6228 Sep 27 22:10 vstafs_stage1_5
-rw-r--r--  1 root root   8764 Sep 27 22:10 xfs_stage1_5
# 看！档案都安装进来了！但是注意到，我们并没有设定档喔！那要自己建立！
```

所以说， grub-install 是安装 grub 到你的装置上面，但是，还需要设定好设定档 (menu.lst) 后， 再以 grub shell 来安装 grub 到 MBR 或者是 Super block 里面去喔！好了，那我们来思考一下想要安装的数据。 鸟哥的 Linux 主机上面，其实仅有一个 Linux 系统，但我的 FC4 已经升级过很多次，所以我的 Linux 有『很多核心』，我想让每个核心都能够使用来开机，而且，还想要将 grub 同时安装在 MBR 与 Super block 当中， 并且 MBR 的 grub 可以将 loader 的控制权转交给 super block ，那么该如何安装呢？ 基于这样的想法，我的设定档应该是这样的：

```
[root@linux ~]# vi /boot/grub/menu.lst
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title Fedora Core (2.6.12-1.1456_FC4)
        root (hd0,0)
```

```
        kernel /boot/vmlinuz-2.6.12-1.1456_FC4 ro root=LABEL=/  vga=787
        initrd /boot/initrd-2.6.12-1.1456_FC4.img
title Fedora Core (2.6.11-1.1369_FC4)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.11-1.1369_FC4 ro root=LABEL=/  vga=787
        initrd /boot/initrd-2.6.11-1.1369_FC4.img
title Fedora Super block loader
          root (hd0,0)
          chainloader +1
```

然后再开始以 grub shell 来进行安装！整个安装与 grub shell 的动作其实很简单， 如果您有兴趣研究的话，可以使用 info grub 去查阅～鸟哥这里仅介绍几个有用的指令而已。

- 用『 root (hdx,x) 』选择含有 /boot 目录的那个 partition 代号；
- 用『 find /boot/grub/stage1 』看看能否找到安装信息档案；
- 用『 find /boot/vmlinuz 』看看能否找到 kernel file (不一定要成功！)；
- 用『 setup (hdx,x) 』或『 setup (hdx) 』将 grub 安装在 super block 或 MBR；
- 用『 quit 』来离开 grub shell ！

所以，请用 grub 来进入 grub shell 吧！进入 grub 后，会出现一个『 grub> 』的提示字符啊！

```
[root@linux ~]# grub

1. 先设定一下含有 /boot 目录的那个 partition 啊！
grub> root (hd0,0)
 Filesystem type is ext2fs, partition type 0x83
# 瞧！找到啦！有这个 partition 的存在，且 grub 认识他为 ext2 的 filesystem。

2. 搜寻一下，是否存在 stage1 这个信息档案？
grub> find /boot/grub/stage1
 (hd0,0)
 (hd1,0)
# 呵呵！竟然找到两个？因为刚刚我们也安装一个在 /dev/hdb1 嘛！

3. 搜寻一下是否可以找到核心？ /boot/vmlinuz ?
grub> find /boot/vmlinuz
Error 15: File not found
grub> find /boot/vmlinuz-2.6.12-1.1456_FC4
 (hd0,0)
# 没办法，FC4 没有连结档，所以需要填写完整的 kernel 文件名称！

4. 给他安装上去吧！安装到 MBR 看看！
grub> setup (hd0)
 Checking if "/boot/grub/stage1" exists... yes
 Checking if "/boot/grub/stage2" exists... yes
```

```
 Checking if "/boot/grub/e2fs_stage1_5" exists... yes
 Running "embed /boot/grub/e2fs_stage1_5 (hd0)"...  15 sectors are embedded.
succeeded
 Running "install /boot/grub/stage1 (hd0) (hd0)1+15 p (hd0,0)/boot/grub/stage2
/boot/grub/grub.conf"... succeeded
Done.
# 很好！确实有装起来～这样 grub 就在 MBR 当中了！

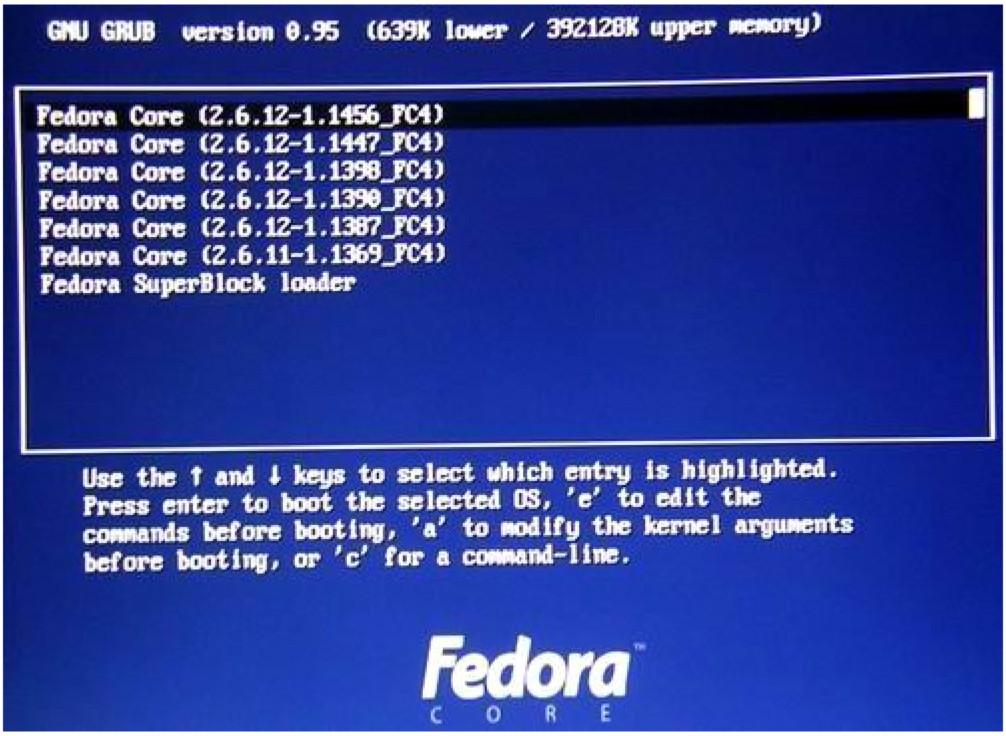
5. 那么重复安装到我的 /dev/hda1 呢？亦即是 super block 当中？
grub> setup (hd0,0)
 Checking if "/boot/grub/stage1" exists... yes
 Checking if "/boot/grub/stage2" exists... yes
 Checking if "/boot/grub/e2fs_stage1_5" exists... yes
 Running "embed /boot/grub/e2fs_stage1_5 (hd0,0)"... failed (this is not fatal)
 Running "embed /boot/grub/e2fs_stage1_5 (hd0,0)"... failed (this is not fatal)
 Running "install /boot/grub/stage1 (hd0,0) /boot/grub/stage2 p
/boot/grub/grub.conf "... succeeded
Done.
# 虽然无法将 stage1_5 安装到 super block 去，不过，还不会有问题，
# 重点是最后面那个 stage1 要安装后，显示 succeeded 字样就可以了！

grub> quit
```

如此一来，就已经将 grub 安装到 MBR 及 super block 里面去了！ 而且读取的是 (hd0,0) 里面的 /boot/grub/menu.lst 那个档案喔！ 真是很重要啊！重要到不行！

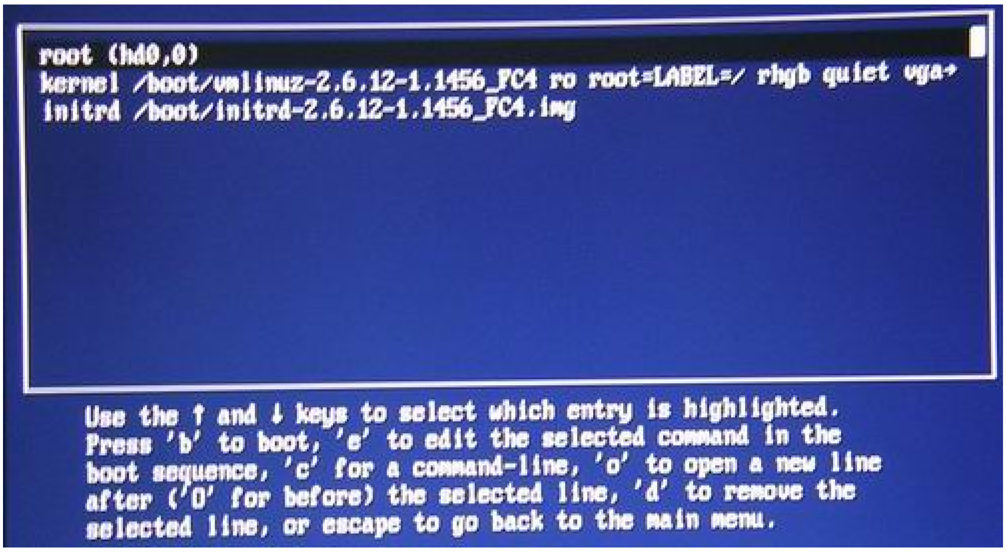
开机前的额外功能修改

事实上，上一个小节设定好之后，您的 grub 就已经在你的 Linux 系统上面了，而且同时存在于 MBR 与 Super block 当中呢！所以，我们已经可以进行重新开机来查阅查阅看看啦！ 另外，如果你正在进行开机，那么请注意，我们可以在预设选单 (鸟哥的范例当中是 5 秒) 按下任意键， 还可以进行 grub 的「在线编修」 功能喔！真是棒啊！先来看看开机画面吧！



图二、 grub 的开机图示

帅吧！鸟哥的主机上面竟然有七个可开机的选单呢！当然啦！要看到这样的选单，你必须要在开机的过程中， 五秒内就得要按下任意键，否则就会进入到正常的开机程序当中了。 这个时候，注意看到上图当中的最底下的一些文字说明，其实，我们可以进行在线编修喔！ 在图二当中，如果我在第一个开机选单当中按下『 e 』这个按键，就会进入 grub shell 的修改， 有点像底下这样：



图三、 grub 的编修画面

这个时候，我可以使用上下键移动光标到想要修改的那一行，然后注意看到图三画面最底下的一些说明文字， 可以使用：

- e：进入 grub shell 的编辑画面；

- o：在游标所在行底下再新增一行；
- d：将游标所在行删除。

我们说过， grub 是可以直接使用核心档案来开机的，所以，如果您很清楚的知道你的根目录 (/) 在那个 partition ，而且知道你的核心档案档名 (通常都会有个 /boot/vmlinuz 连结到正确的档名)， 那么直接在图三的画面当中，以上述的 o, d, e 三个按键来编修，成为类似底下这样：

```
grub edit> kernel (hd0,0)/boot/vmlinuz root=/dev/hda1
```

按下 [ESC] 按键后，然后输入 b 来 boot ，就可以开机啦！所以说， 万一你的 /boot/grub/menu.lst 设定错误，或者是因为安装的缘故， 或者是因为核心档案的缘故，导致无法顺利开机时，记得啊，可以在 grub 的选单部分， 使用 grub shell 的方式去查询 (find) 或者是直接指定核心档案，就能够开机啦！ ^_^

另外，我们刚刚图二画面当中的最后一个选项不是指定到 Super block 吗？ 如果你选择那个项目开机会怎样？哈哈！立刻又进入 grub 的画面当中！ 因为此时 grub 是 super block 当中的，而不是 MBR 当中的！ 如此一来，您就应该会了解到 loader 控制权的转移了吧？也能够知道如何制作多重开机了吧？ 呼呼！加油的啦！

另外，很多时候我们的 grub 可能会发生错误，导致『连 grub 都无法启动』，那么根本就无法使用 grub 的在线编修功能嘛！怎么办？没关系啊！我们可以利用具有 grub 开机的 CD 来开机， 然后再以 CD 的 grub 的在线编修，嘿嘿！同样可以使用硬盘上面的核心档案来开机啦！ 很好玩吧！ ^_^

关于核心功能当中的 vga 设定：

或许刚刚我们在前几个小节提到 menu.lst 内的 kernel 设定当中，你就看到这样的一行：『 kernel /boot/vmlinuz ro root=/dev/hda1 vga=771 』怪怪～那个 771 是啥玩意儿？ 没有他可不可以啊？当然可以啊！只是这个 vga 的设定项目主要功能用来：『 设定终端机 tty1~tty6 的分辨率与色彩度 』啦！ 他的十进制代码与相对应的分辨率与彩度为：

彩度\分辨率	640x480	800x600	1024x768	1280x1024	bit
256	769	771	773	775	8 bit
32768	784	787	790	793	15 bit
65536	785	788	791	794	16 bit
16.8M	786	789	792	795	32 bit

不过，某些操作系统支持的是 16 进位制，所以还需要修改一下格式呢！一般使用上表当中的值应该就可以了。 鸟哥我的屏幕是 17 吋的，所以我是将终端机分辨率调整成 800x600 ，使用 vga=787 就绰绰有余啰～^_^ 不过，由于不同的操作系统与硬件可能会有不一样的情况，因此， 上面的值不见得一定可以在您的机器上面测试成功， 建议您可以分别设定看看哩～以找出可以使用的值！ ^_^

关于大硬盘的问题

虽然我们前面讲过 grub 已经克服了核心放置在 1024 磁柱以后的问题， 不过，如果主机板还是不支持大硬盘装置，那么，嘿嘿嘿嘿！ 可能还是会无法启动 Linux 喔！他会一直告诉你，有 error 18 产生～ 实际的代号可以到底下查询：

- http://orgs.man.ac.uk/documentation/grub/grub_toc.html#SEC_Contents

解决的方法则如同底下两篇讲的：

- http://wiki.linuxquestions.org/wiki/GRUB_boot_menu
- http://forums.gentoo.org/viewtopic.php?t=122656&highlight=grub+error+collection

由于 Linux 核心只要能够被加载到内存当中，那么他就可以自行侦测硬件， 而不以 BIOS 侦测的硬件结果来执行 Linux 的。所以啰，只要能够加载 Linux kernel ，那就万事 OK 了～ 所以，虽然你的主机板不认识大于 120GB 以上的硬盘，但是 Linux 依旧可以使用他。

可惜的是，这个大前提是『 Linux kernel 可以被加载到系统当中』才行～ 但是， BIOS 都读不到核心档案了，该如何载入啊！因此，如果你的 / 切的太大， 偏偏又没有制作 /boot 的 partition ，同时主机板又不支持大硬盘，哈哈哈哈！ 那么首次安装完成之后，就会直接跑到 grub> 的画面当中， 是没有办法进入 Linux 的啦！

在这样的情况下，你可以有一个最简单的做法，就是，直接重灌， 并且制作出 /boot 挂载的 partition ，同时确认该 partition 是在 1024 cylinder 之前才行。 如果实在不想重灌，没有关系，利用我们刚刚上头提到的 grub 功能， 额外建立一个可开机软盘，或者是直接以光驱开机，然后以 grub 的编写能力进入 Linux 。

当然，最好的办法其实是骗过 BIOS ，直接将硬盘的 cylinder, head, sector 等等信息直接写到 BIOS 当中去，如此一来，嘿嘿嘿嘿！你的 BIOS 可能就可以读得到与支持的到你的大硬盘了。 不过，鸟哥还是建议您可以重新安装，并且制作出 /boot 这个 partition 啦！ ^_^

Boot Loader: LILO

说实在的，整个开机以 grub 来作为 boot loader 就很棒了～没有什么需要玩 LILO 的啦！ 不过， grub 还是有点小缺点的，那就是，当你的 partition 变了，或者是 Windows 存在， 但 Linux 死掉时，因为在 Linux 的设定档 (/boot/grub/menu.lst) 挂点， 将会导致无法启动 Windows 的困境，除非您很清楚的知道如何使用 grub shell ，否则， 还真糗～

在这个部分， LILO 则使用与 grub 不同的机制，他将 boot loader 的 stage1 与 stage2 通通写入 MBR 或者是 super block 开机区当中，所以，设定档当然就不需要一定要存在于 Linux 的 filesystem 上啰！ 您说对吧！ ^_^。 但还是各有利弊得失啦～没有说那个比较好就是了。但也因为如此，所以： 当 LILO 设定档被改过后，一定需要重新安装 LILO 一次。 这一点与 grub 是完全不同的呢。

其实，LILO (LInux LOader) 看名称就知道是 Linux 最早的 boot loader ， 他主要利用 /etc/lilo.conf 这个设定档，然后再以 lilo 主程序将该设定内容写入开机区当中。 接下来我们就来玩一玩 LILO ，不过需要留意的是， FC4 似乎没有提供 Lilo 给我们呢！ 所以使用 FC4 的朋友应该就没有办法玩这个咚咚了～不过没关系吧～知道 boot loader 即可啊！ ^_^

LILO 的设定档 /etc/lilo.conf：

LILO 的设定档 /etc/lilo.conf 同样的分为两部分，分别是 LILO 整体环境设定部分， 与每个开机项目核心文件名称规范部分。有点像这样啦：

```
[root@linux ~]# vi /etc/lilo.conf
# 第一部份，整体的设定部分
prompt      <==强制出现 boot 的开机讯息啰！
Compact     <==可以整合一些读取的扇区，可以保持 map 较小，适合软盘开机时使用
timeout=50  <==如果有多重开机的话，可以设定这个延迟时间，单位 0.1 秒
default=linux-2.4.18  <==预设的开机项目，与底下的 label 对应！
boot=/dev/hda         <==Lilo 的开机信息写入到 /dev/hda 这颗硬盘的 MBR 当中。
map=/boot/map         <==用来说明 local 主机的地图信息啰！
install=/boot/boot.b  <==关于开机区的讯息(boot sector)，不用理他没关系！
Linear                <==在较大容量的硬盘使用时，可以加入这一个参数试试看！
lba32                 <==这个东西也是在大容量的硬盘使用时候会需要的参数！
password=1234567      <==设定密码！如果为了安全起见，可以设定您的 lilo 密码哩！
message=/boot/message <==那个 LILO 的讯息就是在里面出现的啦！

# 第二部分，个别的开机设定部分，一个 image 或 other 均代表一个开机设定！
image=/boot/vmlinuz-2.4.7-10     <==核心档案啦！
        label=linux-2.4.7        <==请注意！label 前面以 [tab] 按键来作为分隔！
        initrd=/boot/initrd-2.4.7-10.img
        read-only                <==开机扇区挂载为只读！
        root=/dev/hda1           <==挂载成 / 这个 root 目录的磁盘！
other=/dev/hdb1                  <==如果是『非 Linux 核心』就以 other 来设定
        label=Windows2k          <==同样的要有 label 来表示这个开机扇区的名称！
```

注意上面的几个项目，在整体环境设定项目当中，要注意：

- timeout=50
  timeout 的设定是 0.1 秒，所以 delay=50 表示延迟时间为 5 秒！
- linear 与 lba32
  linear 与 lba32 通常用在 SCSI 或者是较大的硬盘，例如磁柱(cylinder)总数超过 1024 磁柱的硬盘，可以使用这个项目来除错！不过，如果是小于 8GB 的硬盘， 这两个东西有没有设定就没有什么影响了！早期的硬盘容量不大，所以 cylinder 不会超过 1024 ，但较新的硬盘容量太大了，如果核心档案 (/boot/vmlinuz) 放置在 1024 磁柱以后，则可能会发生无法读取的问题，因此需要设定这个 lba32 啊！ 这也是为何很多 distribution 预设都会将 /boot 独立出来的缘故！
- default
  default 需要设定成底下几个 image 或者是 other 的 label 才成！ 这个地方最常被忘记！因为常常会记得修改 label ，但是忘记跟着改变 default 的内容！此外，如果你想要修正开机预设的操作系统选项，在这里改啦！
- password
  password 的用途在于安全防护方面，不过有个困扰，就是『 如果你的计算机因为不正常关机 (如

断电后重开）而在电源恢复的时候重新开机时，则会卡在这个阶段无法直接进入 Linux 系统』，因为你必须提供 password 才能继续的工作呀！

- boot
  boot 显示的是开机的扇区选择！ 这里也蛮重要的，如果你想要安装在 MBR 里面的话，如同上面的书写模式，就是写入 /dev/hda ，后面不要加上每个 partition 的代码！但是，如果你是想写入 Super Block ，例如我想要写入的是 hda5 这个 Logical 的 partition 时，那么这里就必需要改写为 /dev/hda5 啰！所以，您应该只要一看到这个 boot 后面接的内容，就会知道那个安装的扇区是 MBR 还是 Super Block 啰！

至于每个开机选单项目的内容，主要分为：

- image=kernel_file
  image 后面就是接核心档案的档名就是了！这主要是针对 Linux 来作的设定啦！ 在 image 底下还有很多的设定项目，每个项目都以 <tab> 按键来缩排，主要的项目有：
  - label：项目名称！出现选单时可以选择的项目；
  - initrd：后面就是接 RAM Disk 的 initrd 档案档名啊！
  - root：这个重要！就是根目录 (/) 的装置代号啊！
  - append：核心额外的功能增加的地方！与 grub 的 kernel 后面接的参数意思很相近！

- other=device
  其实这个就是 chain loader 啦！移交 boot loader 控制权的设定项目。 在 other 后面接的就是磁盘的装置代号，不论是 MBR 或 super block 都可以啊！里面只要有 label 即可啊！

大致就是这样啊～如果还有什么疑问，详情请参考 man lilo.conf 即可呦！

测试与安装 LILO 开机管理程序：

好啦！为了测试一下您是否已经知道了 lilo.conf 的设定方式，所以我们来做个实验吧！ 请在您的『实验主机』上面，不要在提供服务的主机上面完呦！否则死掉了不要怪我没警告您… 我们先试图安装在 super block 上面好了！以下面为例，特殊字体的部分是经过我的修改之后的结果， 您的 /etc/lilo.conf 应该会长的跟我的差不多才是！

```
[root@linux ~]# vi /etc/lilo.conf
boot=/dev/hda1
map=/boot/map
vga=normal
default=linux
keytable=/boot/us.klt
prompt
nowarn
timeout=100
message=/boot/message
menu-scheme=wb:bw:wb:bw
```

```
image=/boot/vmlinuz
        label=linux
        root=/dev/hda1
        initrd=/boot/initrd.img
        append="devfs=mount"
        read-only
image=/boot/vmlinuz
        label=failsafe
        root=/dev/hda1
        initrd=/boot/initrd.img
        append="devfs=nomount failsafe"
        read-only
image=/boot/vmlinuz       <==就给他新增加一个 label ，但是内容不变!
        label=linux-test
        root=/dev/hda1
        initrd=/boot/initrd.img
        append="devfs=mount"
        read-only
```

这样就设定好了！接着下来看一下怎么安装他吧！安装真是简单到不行～直接输入 lilo 即可！

```
[root@linux ~]# lilo
Added linux *       <==有打星号的是『预设的开机设定档！』
Added failsafe
Added linux-test
```

看到没有！要像上面这样才是安装成功呦！如果出现了错误的讯息，那么肯定是有地方没有安装好！ 这个时候请特别的再重新设定一次 /etc/lilo.conf 呢！有打星号的是『预设的开机设定档！』 而如果您还要看看更多的讯息，那么就需要这样：

```
[root@linux ~]# lilo -v
LILO version 22.3.2, Copyright (C) 1992-1998 Werner Almesberger
Development beyond version 21 Copyright (C) 1999-2002 John Coffman
Released 11-Jul-2002 and compiled at 21:48:42 on Aug 13 2002.

Reading boot sector from /dev/hda1
Using MENU secondary loader
Calling map_insert_data
Mapping message file /boot/message -> message-text
Calling map_insert_file

Boot image: /boot/vmlinuz -> vmlinuz-2.4.19-16mdk
Mapping RAM disk /boot/initrd.img -> initrd-2.4.19-16mdk.img
Added linux *

Boot image: /boot/vmlinuz -> vmlinuz-2.4.19-16mdk
```

```
Mapping RAM disk /boot/initrd.img -> initrd-2.4.19-16mdk.img
Added failsafe

Boot image: /boot/vmlinuz -> vmlinuz-2.4.19-16mdk
Mapping RAM disk /boot/initrd.img -> initrd-2.4.19-16mdk.img
Added linux-test

/boot/boot.0301 exists - no backup copy made.
Writing boot sector.
```

如果你需要更多的讯息，那么就使用『 lilo -v -v -v 』多几个 -v 就对了！ ^_^

一些问题的解决之道：

好了！ lilo 安装完成之后，总是会有一些问题会发生吧！那么如何来解决问题呢？嗯！ 可以看一下底下的一些解决之道：

- 我要如何选择不同的开机设定档？开机的时候我只看的到 boot: 而已?

  开机之后，如果是以 lilo 来启动 kernel 时，那么他会出现 boot: 的字样，出现这个字样之后，马上按下 <tab> 按键，那么就会出现目前 lilo 所记忆的开机设定文件啰！然后在 boot 后面输入想要的开机档案，就可以啰！
- 安装好了 Linux 之后，在开机的过程中却只出现『 LI 』就停止了！该如何是好？

  这个问题可能发生的原因是 Lilo 没有设定好，或者是由于 Linux 安装在非 /dev/hda (MBR) 的硬盘之中，解决的方法可以如下：

  1. 用 Linux 光盘开机，然后在出现 boot: 处输入『 linux root=/dev/hda1 （这个与你的 Linux 安装的 partition 有关）』 顺利开机之后，以 vi 修改 /etc/lilo.conf 将『linear』 这一行取消（如果没有这一行的话，那就在 lilo.conf 中加入吧！）然后执行 『lilo』重新安装 Lilo， 再取出光盘并重新开机试试看；
  2. 进入 BIOS ，将硬盘的 mode 改成 LBA 试看看；
  3. 将 Linux 往前面一点的扇区安装，例如你可能安装在 /etc/hdc1 ，那你可以重新安装 Linux 在 /dev/hda2 试看看

- 安装 Linux 完成之后，却是出现 010101... 等数字在屏幕上，无法进入 Linux ...

  这个问题的发生很有可能是硬盘出了问题了！这个时候可以使用 fsck 来扫瞄啰！

  1. 用软盘或者是光盘开机后，使用 fsck 这个硬盘修正软件扫瞄一下您的 root partition ，例如：fsck /dev/hda1
  2. 进入 BIOS ，将硬盘的 mode 改成 LBA 试看看；

- 我们知道 DOS 需要在第一颗硬盘的第一个扇区才能正常开机使用！ 那要是他并非在第一个扇区呢？例如当 DOS 系统在 /dev/hdb1 （第一条排线的 slave ）？

  解决知道就是以 lilo 修正磁盘的配置啦！如下所示来修改 /etc/lilo.conf

  ```
  other=/dev/hdb1
      label=DOS
      map_drive=0x80
      to=0x81
      map_drive=0x81
      to=0x80
  ```

  然后再执行 lilo 写入 MBR 当中！
- 我不要玩 Linux 了，如何移除 lilo ？

  只要以 Windows 的开机片开机，然后以 Windows 系统的 fdisk 下达：『fdisk /mbr』 就可以将 Lilo 自 MBR 当中移除啰！
- 无法正常的进入 Lilo 怎么办？

  这的时候开机片就很重要啦！使用开机片，在出现 boot: 的时候，输入 『linux -s』 就可以进入啦！这里请特别留意！那个 linux 指的是 label 呢！就像刚刚我们有三个 label ，分别是 linux, linux-test 与 failsafe ，那么如果我要以 linux-test 这个开机设定文件的单人维护模式登入，就必需要改写成

  linux-test -s

  linux-test single

  请特别留意！！

开机过程的问题解决：

很多时候，我们可能因为做了某些设定，或者是因为不正常关机（例如未经通知的停电等等）而导致系统的 filesystem 错乱，此时，Linux 可能无法顺利开机成功，那怎么办呢？难道要重灌？当然不需要啦！进入 run level 1（单人维护模式）去处理处理，应该就 OK 的啦！底下我们就来谈一谈如何处理几个常见的问题！

忘记 root 密码的解决之道：

大家都知道鸟哥的记忆力不佳，容易忘东忘西的，那如果连 root 的密码都忘记了，怎么办？ 其实在 FC4 上面，root 密码忘记的情况下，应该是不难解决啦！只要能够进入并且挂载 / ， 然后修改一下 /etc/shadow 内的 root 密码栏（第二栏啊，参考账号管理）重新开机后， root 就不需要密码即可登入啊！因为在 FC4 上面进入 run level 1 是不需要密码的。 整个动作有点像这样：

1. 在开机的时候，到达选单时，我们以较常见的 grub 作为介绍。出现选单后， 将光棒移动到要开机的那个项目上面，然后按下『 e 』进入细项设定，选择『 kernel 』那一项， 再按『 e 』进

入编修画面，在最后面加上一个单一的『 1 』(数字 1, 2, 3 的 1 啊！)， 按下 [Esc] 按键，然后按下『 b 』，就能够以该 kernel 进入 run level 1 了。
2. 进入 Linux 后，不需要输入密码，直接就会是 root 的身份，立刻 vi /etc/shadow ， 将 root 所在那一行的第二个字段给他全部抹除，储存后离开，然后『 reboot 』重新开机。
3. 由于 root 没有密码了，最好在重新开机前就将网络线拔掉，然后以 root 登入，然后立即设定 root 新密码，这样 root 的密码就算是救回来了。

那如果你的 Linux distribution 算是比较严谨的，所以登入 run level 1 时，还是得要输入 root 密码，怎么办？很简单啊！可以：

- 使用 Live CD ，例如 KNOPPIX (可以在台南县网中心，小三老师发起的，阿里巴巴兄负责维护的这个网站： http://knoppix.tnc.edu.tw/ 下载)， 将 KNOPPIX 的映象档下载，然后烧录成为光盘，并以此片光盘开机， 就能够进入 Linux 系统啦！之后，再挂载 / ，然后按照上面的密码修改一下， 嘿嘿！成功！
- 在开机的选单上，将原本 kernel 项目最后方加上『 init=/bin/bash 』修改一下登入的 shell ，不使用 init ，就能够不使用 init ，而直接丢一个 shell 给使用者。不过， 除非很严重的错误，否则不要用这个方法！

因设定错误而无法开机：

如果因为设定错误导致无法开机时，要怎么办啊？这就更简单了！ 最容易出错的设定而导致无法顺利开机的步骤，通常就是 /etc/fstab 这个档案了， 尤其是使用者在 实作 Quota 时，最容易写错参数， 又没有经过 mount -a 来测试挂载，就立刻直接重新开机，真要命，无法开机成功怎么办？ 不要紧啦！利用上个小节提到的以 run level 1 的方法进入 Linux 系统，然后：

- 利用『 mount -n -o remount,rw / 』重新挂载根目录， 之后将刚刚设定错误的地方修改一下，就可以重新开机啦！

但万一是因为不正常关机，导致开机时进行 fsck 无法成功，而出现类似这样的几行字：

```
/home contains a file system with errors,check blocks.
/home:Group 81's inode table at 2654219 conflicts with some other fs blocks.
/home: UNEXPECTED INCONSISTENCY ; RUN fsck MANUSLLY
(i.e. , without -a or -p options)

*** An error occurred during the file system check.
*** Dropping you to a shrll ; the system will reboot
*** when you to leave shell....
 Give root password for maintenance(or type Control-D for normal startup):
```

这表示你的 filesystem 可能有扇区错乱的情况，一般来说，这样的扇区错乱应该不是实体硬盘错误， 比较可能是由于不正成关机造成 filesystem 的不一致 (Inconsistent) 所造成的。 造成这个问题之后，我们必须要输入 root 的密码，进入 run level 1 ， 然后以 fsck /dev/hd[a-d][1-16] 来修复磁盘。例如，假设上面的案例中， /home 挂载在 /dev/hda6 上面，那我就『 fsck /dev/hda6 』，不要加上任何参数。等到系统发现错误，并且出现『clear [Y/N]』时，输入『 y 』吧！

这个过程可能会很长，而且如果你的 partition 上面的 filesystem 有过多的数据损毁时，即使 fsck 完成后，可能因为伤到系统槽，导致某些关键系统档案数据的损毁，那么依旧是无法进入 Linux 的。此时，就好就是将系统当中的重要数据复制出来，然后重新安装，并且检验一下， 是否实体硬盘有损伤的现象才好！不过一般来说，不太可能会这样啦～ 通常都是 fsck 处理完毕后，就能够顺利再次进入 Linux 了。

利用 chroot 切换到另一颗硬盘工作

仔细检查一下，你的 Linux 里面应该会有一个名为 chroot 的指令才对！这是啥？ 这是『 change root directory 』的意思啦！意思就是说，可以暂时将根目录移动到某个目录下， 然后去处理某个问题，最后再离开该 root 而回到原本的系统当中。

举例来说，补习班中心最容易有两三个 Linux 系统在同一个主机上面，假设我的第一个 Linux 无法进入了，那么我可以使用第二个 Linux 开机，然后在第二个 Linux 系统下将第一个 Linux 挂载起来， 最后用 chroot 变换到第一个 Linux ，就能够进入到第一个 Linux 的环境当中去处理工作了。

你同样也可以将你的 Linux 硬盘拔到另一个 Linux 主机上面去，然后用这个 chroot 来切换， 以处理你的硬盘问题啊！那怎么做啊？粉简单啦！

1. 用尽任何方法，进入一个完整的 Linux 系统 (run level 3 或 5)；
2. 假设有问题的 Linux 磁盘在 /dev/hdb1 上面，且他整个系统的排列是：

   / → /dev/hdb1
   /var → /dev/hdb2
   /home → /dev/hdb3
   /usr → /dev/hdb5

   若如此的话，那么在我目前的这个 Linux 底下，我可以建立一个目录， 然后可以这样做：

   /chroot/ → /dev/hdb1
   /chroot/var/ → /dev/hdb2
   /chroot/home/ → /dev/hdb3
   /chroot/usr/ → /dev/hdb5

   全部挂载完毕后，再输入『 chroot /chroot 』嘿嘿！你就会发现，怎么根目录 (/) 变成那个 /dev/hdb1 的环境啦！这样说明，瞭了吗？ ^_^

参考数据

- info grub
- GNU 官方网站关于 grub 的说明文件：
  http://www.gnu.org/software/grub/manual/html_node/

- 纯文字屏幕分辨率的修改方法：
  http://phorum.study-area.org/viewtopic.php?t=14776

本章习题练习

（ 要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看 ）

- 如何察看与修改 runlevel 呢？

  察看很简单，只要输入『 runlevel 』就可以得知。而如果要修改目前的 runlevel ， 可以直接输入 init [level] 例如要去到 runlevel 3 可以：『 init 3 』即可。 如果想要每次开机都设定固定的 runlevel ，那么可以修改 /etc/inittab 这个档案！ 将里面这一行改成：
  『id:3:initdefault:』即可。

- 我有个朋友跟我说，他想要让一个程序在 Linux 系统下一开机就启动，但是在关机前会自动的先结束该程序， 我该怎么建议他？

  由于 /etc/rc.d/rc[0-6].d 里面有的 Sxxname 与 Kxxname 可以设定开机启动与关机结束的事项!所以我就可以轻易的写一个 script 放在 /etc/rc.d/init.d 里面,并连结到我的 run-level 里头，就可以让他自由自在的启动与结束了！

- 万一不幸,我的一些模块没有办法让 Linux 的核心捉到,但是偏偏这个核心明明就有支持该模块,我要让该模块在开机的时候就被加载，那么应该写入那个档案？

  应该写入 /etc/modprobe.conf (kernel 2.6.x) 或者是 /etc/modules.conf (kernel 2.4.x) 这个档案，他是模块加载相关的地方呢！当然，也可以写入 /etc/sysconfig/modules/* 里面。

- 如何在 grub 开机过程当中，指定以『 run level 1 』来开机？

  在开机进入 boot loader 之后，利用 grub shell 的功能，亦即输入『 e 』进入编辑模式， 然后在 kernel 后面增加：
  kernel (hd0,0)/boot/vmlinuz ro root=/dev/hda1 .... single
  那个 single 也可以改成 1 ，就能够进入。同样的，若使用 lilo 时，按下 tab 按键后， 输入
  label_name -s
  就能够进入 run level 1 啰！

- 由于一些无心之过，导致系统开机时，只要执行 init 就会产生错误而无法继续开机， 我们知道可以在开机的时候，不要以 init 加载系统，可以转换第一支执行程序， 假设我第一支执行程序想要改为 /bin/bash ，好让我自行维护系统(不同于 run level 1 喔!)， 该如何进行此一工作？

  在开机的过程当中，进入 lilo 或 grub 的画面后，在 kernel 的参数环境下，加入
  init=/bin/bash 来取代 /sbin/init ，则可略过 init 与 /etc/inittab 的设定项目，不过，您必须相当熟悉 grub 与 lilo 的设定才行喔！ ^_^

- 在 FC4 当中，我们如何自动可加载的模块？

可以经由设定 /etc/modprobe.conf 或者是将自行做好的设定文件写入到 /etc/sysconfig/modules/ 目录中， 并且将档名取为 filename.modules ，注意喔，档案结果务必是 .modules 才行。 相关信息可以参考 /etc/rc.d/rc.sysinit 喔！

原始码与 Tarball 套件管理员

最近更新日期：2005/10/01

我们在 Linux 是什么 一文当中，提到了 GNU 与 GPL 还有开放源码等咚咚，不过，前面都还没有提到真正的开放源码是什么的讯息！在这一章当中，我们将藉由 Linux 操作系统里面的执行文件，来理解什么是可执行的程序，以及了解什么是编译器。 另外，与程序息息相关的函式库 (library) 的信息也需要了解一番！不过， 在这个章节当中，鸟哥并不是要您成为一个开放源码的程序设计师， 而是希望您可以了解如何将开放源码的程序设计、加入函式库的原理、透过编译而成为可以执行 的 binary file ，最后该执行档可被我们所使用的一连串过程！

了解上面的咚咚有什么好处呢？！因为在 Linux 的世界里面，我们常常需要自行安装套件在自己的 Linux 系统上面，所以如果您有简单的程序编译概念，那么将很容易进行套件的安装， 甚至在发生套件编译过程中的错误时，您也可以自行作一些简易的修订呢！而最传统的套件安装过程， 自然就是由原始码编译而来的啰！所以，在这里我们将介绍最原始的套件管理方式：使用 Tarball 来安装与升级管理我们的套件喔！

前言

如果鸟哥想要在我的 Linux 服务器上面跑网页服务器(WWW server)这项服务，那么我应该要做些什么事呢？呵呵！当然就一定需要『 安装网页服务器的套件』啰！ 如果鸟哥的服务器上面没有这个套件的话，那当然也就无法启用 WWW 的服务啦！所以啦，想要在您的 Linux 上面进行一些有的没的功能，学会『如何安装套件』是很重要的一个课题！

咦！安装套件有什么难的？在 Windows 操作系统上面安装套件时，不是只要一直给他按 『下一步』就可以安装妥当了吗？话是这样说没错啦，不过，也由于如此，所以在 Windows 系统上面的软件都是一模一样的，也就是说，您『 无法修改该软件的原始程序代码』，因此， 万一您想要增加或者减少该软件的某些功能时，呵呵！大概只能求助于当初发行该软件套件的厂商了！

或许你会说：『唉呦！我不过是一般人，不会用到多余的功能，所以不太可能会更动到程序代码的部分吧！？』如果您这么想的话，很抱歉～是有问题的！怎么说呢？ 像目前网络上面的病毒、黑客软件、臭虫程序等等，都可能对您的主机上面的某些软件造成影响 (这是因为软件开发者在写作之初可能并没有想到某些问题所致！)，导致主机的当机或者是其它数据损毁等等的伤害。如果您可以藉由安全信息单位所提供的修订方式进行修改， 那么您将可以很快速的自行修补好该软件的漏洞，而不必一定要等到套件开发商提供修补的程序包哩！要知道， 提早补洞是很重要的一件事。

这样说可以了解 Linux 的优点了吗？！没错！因为 Linux 上面的套件几乎都是经过 GPL 的授权，所以每个套件几乎均提供原始程序代码， 并且您可以自行修改该程序代码，以符合您个人的需求呢！很棒吧！这就是开放源码(Open source)的优点啰！不过，到底什么是开放源码？这些程序代码是什么咚咚？又 Linux 上面可以执行的相关套件档案与开放源码之间是如何转换的？不同版本的 Linux 之间能不能使用同一个执行档？或者是该执行档需要由原始程序代码的部分重新进行转换？ 这些都是需要厘清观念的。底下我们先就原始程序代码与可执行档来进行说明。

什么是开放源码、 编译器与可执行档？

在讨论程序代码是什么之前，我们先来谈论一下什么是可执行档？我们说过，在 Linux 系统上面，一个档案能不能被执行看的是有没有可执行的那个权限(具有 x permission)，不过，Linux 系统上真正认识的可执行文件其实是二进制档案(binary file)，例如 /usr/bin/passwd, /bin/touch 这些个档案即为 binary 的可执行档案！

或许您会说，咦！shell scripts 不是也可以执行吗？!其实 shell scripts 只是利用 shell (例如 bash)这支程序的功能进行一些判断式，而最终执行的除了 bash 提供的功能外，仍是呼叫一些已经编译好的 binary 档案来执行的呢！(bash 本身就是 binary file 喔！) 那么我怎么知道一个档案是否为 binary 呢？！还记得我们在 Linux 档案与目录管理 里面提到的 file 这个指令的功能吗？！对啦！用他就是了！我们现在来测试一下：

```
# 先以系统的档案测试看看：
[root@linux ~]# file /bin/bash
/bin/bash: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.2.5, dynamically linked (uses shared libs), stripped
```

```
# 如果是系统提供的 /etc/init.d/syslog 呢?
[root@linux ~]# file /etc/init.d/syslog
/etc/init.d/syslog: Bourne-Again shell script text executable
```

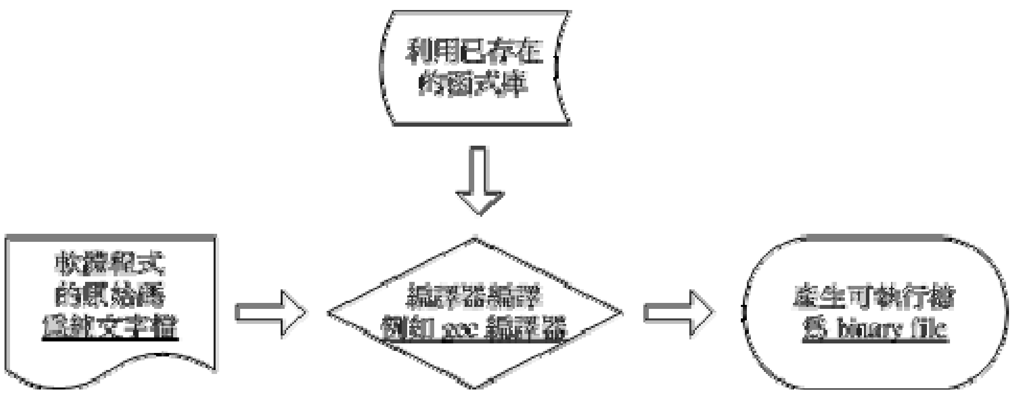
看到了吧！如果是 binary file 而且是可以执行的时候，他就会显示执行文件类别 (ELF 32-bit LSB executable)，同时会说明是否使用动态函式库(shared libs)，而如果是一般的 script ，那他就会显示出 text executables 之类的字样！

> Tips:
> 事实上，syslog 的数据显示出 Bourne-Again ... 那一行，是因为您的 scripts 上面第一行有宣告 #!/bin/bash 的缘故，如果您将 script 的第一行拿掉，呵呵！那么不管 /etc/init.d/syslog 的权限为何，他其实显示的是 ASCII 文字文件的信息喔！



既然 Linux 操作系统真正认识的其实是 binary file ，那么我们是如何做出这样的一支 binary 的程序呢？！首先，我们必须要写程序，用什么东西写程序？就是一般的文书处理器啊！我都喜欢使用 vi 来进行程序的撰写，写完的程序就是所谓的原始程序代码啰！ 这个程序代码档案其实就是一般的纯文字文件 (text file)。 在完成这个原始码档案的编写之后，再来就是要将这个档案『 编译』成为操作系统看的懂得 binary file 啰！而要编译自然就需要『编译器』来动作， 经过编译器的编译之后，就会产生一支可以执行的 binary file 啰。

举个例子来说，在 Linux 上面最标准的程序语言为 C ，所以我使用 C 的语法进行原始程序代码的书写，写完之后，以 Linux 上标准的 C 语言编译器 gcc 这支程序来编译，就可以制作一支可以执行的 binary file 啰。整个的流程有点像这样：



图一、简易的 gcc 编译流程

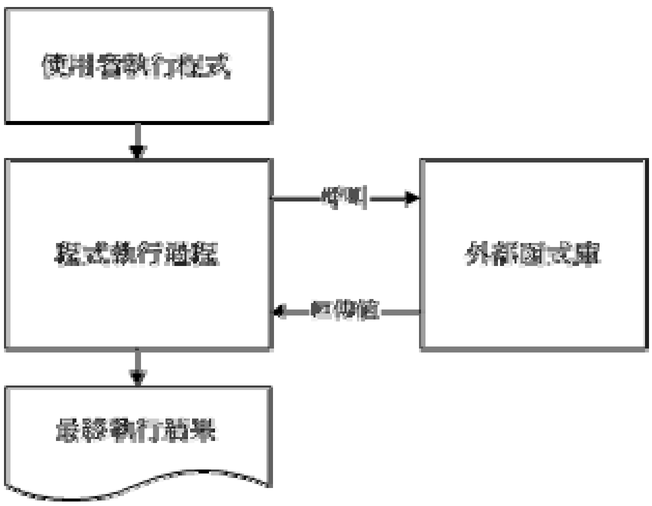
事实上，在编译的过程当中，还会产生所谓的目标文件(Object file)，这些档案是以 *.o 的附文件名样式存在的！至于 C 语言的原始码档案通常以 *.c 作为附档名。此外，有的时候，我们会在程序当中『引用、呼叫』 其它的外部子程序，或者是利用其它套件提供的『函数功能』， 这个时候，我们就必须要在编译的过程当中，将该函式库给他加进去，如此一来， 编译器就可以将所有的程序代码与函式库作一个连结(Link)以产生正确的执行档啰。

什么是函式库?

在前一小节的图一示意图中，在编译的过程里面有提到函式库。好啦， 那么什么是函式库呢？先举个例子来说，我们的 Linux 系统上，系统通常已经提供一个可以进行身份验证的模块，称为 PAM ，这个 PAM 提供的功能可以让很多的程序在被执行的时候，除了可以验证使用者登入的信息外， 还可以将身份确认的数据记录在登录文件(log file, 请查阅后续的 认识登录档 一文)里面，以方便系统管理员的追踪!

既然有这么好用的功能，那如果我要编写具有身份认证功能的程序时， 直接引用该 PAM 的功能就好啦，如此一来，我就不需要重新设计认证机制啰! 也就是说， 只要在我写的程序代码里面，设定去呼叫 PAM 的函式功能，呵呵!我的程序就可以利用 Linux 原本就有的身份认证的程序咯!除此之外，其实我们的 Linux 核心(kernel)也提供了相当多的函式库来给硬件开发者利用喔。

函式库又分为动态与静态函式库，这两个咚咚的分别我们在后面的章节再加以说明。 这里我们以一个简单的流程图，来示意一支有呼叫外部函式库的程序的执行情况。



图二、程序引用函式库的示意图

很简单的示意图啊! ^_^! 而如果要在程序里面加入引用的函式库，就需要如图一所示， 亦即在编译的过程当中，就需要加入函式库的相关设定啰。

事实上， Linux 的核心提供很多的核心相关函式库与外部参数， 这些核心功能在设计硬件的驱动程序的时候是相当有用的信息，这些核心相关信息大多放置在 /usr/include, /lib, /usr/lib 里面哩! 我们在本章的后续小节再来探讨。

什么是 make 与 configure ?

事实上，使用类似 gcc 的编译器来进行编译的过程并不简单， 因为除了每个主程序与子程序均需要写上一笔编译过程的指令外，还需要写上最终的连结程序。 程序代码小的时候还好，如果是类似 WWW 服务器软件(例如 Apache)，或者是类似核心的原始码，动则数百 MBytes 的资料量，呵呵! 指令会写到疯掉～ 这个时候，我们就可以使用 make 这个指令的相关功能来进行编译过程的指令简化了!

当执行 make 时，make 会在当时的目录下搜寻 Makefile (or makefile) 这个文字文件，而 Makefile 里

面则记录了原始码如何编译的详细信息！ make 会自动的判别原始码是否经过变动了，而自动更新执行档，是软件工程师相当好用的一个辅助工具呢！

咦！make 是一支程序，会去找 Makefile ，那 Makefile 怎么写？呵呵！ 通常软件开发商都会写一支侦测程序来侦测使用者的作业环境， 以及该作业环境是否有软件开发商所需要的其它功能，该侦测程序侦测完毕后，就会主动的建立这个 Makefile 的规则档案啦！通常这支侦测程序的文件名为 configure 或者是 config 。

咦！？那为什么要侦测作业环境呢？！在 什么是 Linux 那个章节当中， 不是有提到 Linux 不过就是一个核心吗？！是这样没错啦！但是您必须要了解的是， 某些软件套件需要一些相关的套件辅助，并且，某些驱动程序则是适用在不同的核心系统(因为核心提供的函式库可能并不相同，例如 kernel 2.4.xx 与 kernel 2.6.xx 就不太一样！)，并且每个 Linux distribution 所提供的函式库名称与路径可能也不太一样，所以说，在 Fedora Core 4 上面可以执行的一个 binary file ，直接复制到 SuSE 平台上，可不见得可以顺利执行(事实上，是不太可能可以执行啦！^_^)。 所以啦，原始码写出来之后，需要针对不同的作业环境来进行编译的行为呐！这个时候就很需要 configure 以及 make 的功能啊！

详细的 make 用法与 Makefile 规则，在后续的小节里面再探讨啰！

什么是 Tarball 的套件 ？

从前面几个小节的说明来看，我们知道所谓的原始程序代码，其实就是一些写满了程序代码的纯文本文件。那我们从前面的 档案的压缩与打包 章节当中，也了解了纯文字文件其实是很浪费硬盘空间的一种档案格式！(想一想，一个核心的原始码档案大约要 200~300 MB 以上，如果每个人都去下载这样的一个核心档案， 呵呵！那么网络频宽不被吃的死翘翘才怪呢！) 所以啦，如果能够将这些原始码透过档案的打包与压缩技术来将档案的数量与容量减小， 不但让使用者容易下载，套件开发商的网站频宽也能够节省很多很多啊！这就是 Tarball 档案的由来啰！

所谓的 Tarball 档案，其实就是将套件的所有原始码档案先以 tar 打包，然后再以压缩技术来压缩，通常最常见的就是以 gzip 来压缩了。因为利用了 tar 与 gzip 的功能，所以 tarball 档案一般的附档名就会写成 *.tar.gz 或者是简写为 *.tgz 啰！也就是说， Tarball 套件解压缩之后，里面的档案通常就会有：

- 原始程序代码档案；
- 侦测程序档案(可能是 configure 或 config 等檔名)；
- 本套件的简易说明与安装说明(INSTALL 或 README)。

其中最重要的是那个 INSTALL 或者是 README 这两个档案，通常您只要能够参考这两个档案，呵呵！Tarball 套件的安装是很简单的啦！我们在后面的章节会再继续介绍 Tarball 这个玩意儿。

如何安装与升级套件

将原始码作了一个简单的介绍，也知道了系统其实认识的可执行档是 binary file 之后，好了，得要聊一聊，那么怎么安装与升级一个 Tarball 的套件？为什么要安装一个新的套件呢？当然是因为我们的主机上面没有该套件啰！那么， 为何要升级呢？！原因可能有底下这些：

- 需要新的功能，但旧有主机的旧版套件并没有，所以需要升级到新版的套件；
- 旧版本的套件上面可能有安全上的顾虑，所以需要更新到新版的套件；
- 旧版的套件执行效能不彰，或者执行的能力不能让管理者满足。

在上面的需求当中，尤其需要注意的是第二点，当一个套件有安全上的顾虑时，千万不要怀疑， 赶紧更新套件吧！否则造成网络危机，那可不是闹着玩的！那么更新的方法有哪些呢？ 基本上更新的方法可以分为两大类，分别是：

- 直接以原始码透过编译来安装与升级；
- 直接以编译好的 binary file 来安装与升级。

上面第一点很简单，就是直接以 Tarball 在自己的机器上面进行侦测、编译、 安装与设定等等动作来升级就是了。不过，这样的动作虽然让使用者在安装过程当中具有很高的弹性， 但毕竟是比较麻烦一点，如果 Linux distribution 厂商能够针对自己的作业平台先进行编译等过程，再将编译好的 binary file 释出的话，那由于我的系统与该 Linux distribution 的环境是相同的，所以他所释出的 binary file 就可以在我的机器上面直接安装啦！省略了侦测与编译等等繁杂的过程呢！

目前很多 binary file 升级的机制呢，包括有 Red Hat 系统 (含 Fedora 系列) 发展的 RPM 与 up2date, yum 等在线更新模式；Mandrake 的 urpmi 套件更新方式； Debian 使用的 dpkg ； Sun Unix 使用的 pkg ，以及目前很流行的 apt 在线更新模式等等， 以使用率来说，目前最传统的是以 Tarball 直接进行编译的安装与升级，而另一个则是以 RPM 相关的机制来进行安装与升级啰！本章节主要针对 Tarball ，至于 RPM 则留待下个章节再来介绍呢！

好了，那么一个套件的 Tarball 是如何安装的呢？基本流程是这样的啦：

1. 将 Tarball 由厂商的网页下载下来；
2. 将 Tarball 解开，产生很多的原始码档案；
3. 开始以 gcc 进行原始码的编译(会产生目标文件 object files)；
4. 然后以 gcc 进行函式库、主、子程序的连结，以形成主要的 binary file；
5. 将上述的 binary file 以及相关的设定文件安装至自己的主机上面。

上面第 3, 4 步骤当中，我们可以透过 make 这个指令的功能来简化他，所以整个步骤其实是很简单的啦！只不过您就得需要至少有 gcc 以及 make 这两个套件在您的 Linux 系统里面才行喔！详细的过程以及需要的套件我们在后面的章节继续来介绍的啦！

一个简单的范例：
经过上面的介绍之后，您应该比较清楚的知道原始码、编译器、函式库与执行档之间的相关性了。 不过，详细的流程可能还是不很清楚，所以，在这里我们以一个简单的程序范例来说明整个编译的过程喔！赶紧进入 Linux 系统，实地的操作一下底下的范例呢！

印出 Hello World

我们以 Linux 上面最常见的 C 语言来撰写第一支程序！第一支程序最常作的就是..... 在屏幕上面印出『Hello World！』的字样～当然， 这里我们是以简单的 C 语言来撰写，如果您对于 C 有兴趣的话，那么请自行购买相关的书籍喔！ ^_^好了，不啰唆，立刻编辑第一支程序吧！ (请先确认您的 Linux 系统里面已经安装了 gcc 了喔！如果尚未安装 gcc 的话，请先参考下一节的 RPM 安装法，先安装好 gcc 之后，再回来阅读本章)

```
# 1. 先编辑原始码：
[root@linux ~]# vi hello.c   <==注意一下， C 语言用 .c 的附檔名
#include <stdio.h>
int main(void)
{
        printf("Hello World\n");
}

# 上面是 C 语言的语法，那个第一行的 # 并不是批注喔！如果您担心
# 输入错误，请到底下的连结下载这个档案：
# http://linux.vbird.org/linux_basic/0520source/hello.c

# 2. 开始将原始码编译成为可执行的 binary file ：
[root@linux ~]# gcc hello.c
# 这个时候，在本目录下会产生一个名为 a.out 的执行档！
# 在预设的状态下，如果我们直接以 gcc 编译原始码，并且没有加上
# 任何参数，则 执行档的档名会被自动设定为 a.out 这个文件名称！

# 3. 执行一下这个档案：
[root@linux ~]# ./a.out
Hello World  <==呵呵！成果出现了！
```

好了，上面的例子很简单吧！那个 hello.c 就是原始码，而 gcc 就是编译器，至于 a.out 就是编译成功的可执行 binary file 啰！咦！如果我想要产生目标文件(object file)来进行其它的动作(在较大的套件当中， 就需要使用多个目标文件来进行相关的连结)，而且执行档的档名也不要用预设的 a.out ，那该如何是好？可以将上面的第 2 个步骤改成这样：

```
[root@linux ~]# gcc -c hello.c
# 会产生一个名为 hello.o 的目标文件，object file 的附档名为 *.o 喔！

[root@linux ~]# gcc -o hello hello.o
# 这个步骤是利用 hello.o 这个目标文件制作出一个名为 hello 的执行档
# 详细的 gcc 语法我们会在后续章节中继续介绍！
# 透过这个动作后，我们可以得到 hello 及 hello.o 两个档案，
# 真正可以执行的是 hello 这个 binary file 喔！

# 3. 执行一下这个档案：
```

```
[root@linux ~]# ./hello
Hello World
```

或许您会觉得，咦！只要一个动作作出 a.out 就好了，干嘛还要先制作目标文件再做成执行档呢？！ 呵呵！透过下个范例，您就可以知道为什么啦！

子程序的编译

如果我们在一个主程序里面又呼叫了另一个子程序呢！？这是很常见的一个程序写法， 因为可以简化整个程序的易读性！在底下的例子当中，我们以 thanks.c 这个主程序去呼叫 thanks_2.c 这个子程序，写法很简单：

```
# 1. 先编辑原始码 thanks.c 与 thanks_2.c：
[root@linux ~]# vi thanks.c
#include <stdio.h>
int main(void)
{
        printf("Hello World\n");
        thanks_2();
}
# 上面的 thanks_2(); 那一行就是呼叫子程序啦！

[root@linux ~]# vi thanks_2.c
void thanks_2(void)
{
        printf("Thank you!\n");
}
# 上面这两个档案您可以到底下下载：
# http://linux.vbird.org/linux_basic/0520source/thanks.c
# http://linux.vbird.org/linux_basic/0520source/thanks_2.c

# 2. 开始将原始码编译成为可执行的 binary file ：
[root@linux ~]# gcc -c thanks.c thanks_2.c
# 产生了两个目标文件！且编译过程里面可能会产生一些 warning (警告) 的讯息，
# 因为仅是警告而已，所以该讯息你可以先略过去不打紧的！ ^_^
[root@linux ~]# gcc -o thanks thanks.o thanks_2.o

# 3. 执行一下这个档案：
[root@linux ~]# ./thanks
Hello World
Thank you!
```

知道为什么要制作出目标文件了吗？！由于我们的原始码档案有时并非仅只有一个档案， 所以我们无法直接进行编译。这个时候就需要先产生目标文件，然后再以连结制作成为 binary 可执行档。另外，如果有一天，您更新了 thanks_2.c 这个档案的内容，则您只要重新编译 thanks_2.c 来产生新的 thanks_2.o ，

然后再以连结制作出新的 binary 可执行档即可！而不必重新编译其它没有更动过的原始码档案。 这对于软件开发者来说，是一个很重要的功能，因为有时候要将偌大的原始码全部编译完成， 会花很长的一段时间呢！

此外，如果您想要让程序在执行的时候具有比较好的效能，或者是其它的除错功能时， 可以在编译的过程里面加入适当的参数，例如底下的例子：

```
[root@linux ~]# gcc -O -c thanks.c thanks_2.c
# -O 为产生最佳化的参数

[root@linux ~]# gcc -Wall -c thanks.c thanks_2.c
thanks.c: In function 'main':
thanks.c:5: warning: implicit declaration of function 'thanks_2'
thanks.c:6: warning: control reaches end of non-void function
thanks_2.c: In function 'thanks_2':
thanks_2.c:3: warning: implicit declaration of function 'printf'
thanks_2.c:3: warning: incompatible implicit declaration of built-in function 'printf'
# -Wall 为产生更详细的编译过程信息。上面的讯息为警告讯息( warning )
# 所以不用理会也没有关系！
```

至于更多的 gcc 额外参数功能，就得要 man gcc 啰～呵呵！可多的跟天书一样～

加入连结的函式库

刚刚我们都仅只是在屏幕上面印出一些字眼而已，如果说要计算数学公式呢？！ 例如我们想要计算出三角函数里面的 sin(90 度角)，要注意的是，大多数的程序语言都是使用径度而不是一般我们在计算的『角度』，180 度角约等于 3.14 径度！嗯！那我们就来写一下这个程序吧！

```
[root@linux ~]# vi sin.c
#include <stdio.h>
int main(void)
{
        float value;
        value = sin ( 3.14 / 2 );
        printf("%f\n",value);
}
# 上面这个档案的内容可以在底下取得！
# http://linux.vbird.org/linux_basic/0520source/sin.c
```

那要如何编译这支程序呢？我们先直接编译看看：

```
[root@linux ~]# gcc sin.c
sin.c: In function 'main':
sin.c:5: warning: incompatible implicit declaration of built-in function 'sin'
/tmp/cciDlilg.o(.text+0x2c): In function `main':
sin.c: undefined reference to `sin'
```

```
collect2: ld returned 1 exit status
# 注意看到上面最后一行，会有个错误讯息，代表没有成功！
```

特别注意上面的错误讯息，唉啊！怎么没有编译成功？它说的是『undefined reference to sin』，说的是『没有 sin 的相关定义参考值！』，为什么会这样呢？这是因为 C 语言里面的 sin 函示是写在 libm.so 这个函式库中，而我们并没有在原始码里面加入相关的说明， 所以当然就需要在编译与连结的时候将这个函式库给他连结进执行档里面啊！所以我们可以这样做：

```
[root@linux ~]# gcc sin.c -lm -L/lib -L/usr/lib
# 特别注意，那个 -lm 可以拆开成两部份来看：
# -l ：是『加入某个函式库(library)』的意思，而
# m  ：则是 libm.so 这个函式库，其中， lib 与附档名(.a 或 .so)不需要写
# 所以 -lm 表示使用 libm.so (或 libm.a) 这个函式库的意思～
# 至于那个 -L 后面接的路径呢？这表示：
#『我要的函式库 libm.so 请到 /lib 或 /usr/lib 里面搜寻！』
[root@linux ~]# ./a.out
1.000000
```

上面的说明很清楚了吧！！不过，要注意的是，由于 Linux 预设是将函式库放置在 /lib 与 /usr/lib 当中，所以您没有写 -L/lib 与 -L/usr/lib 也没有关系的！不过，万一哪天您使用的函式库并非放置在这两个目录下，那么 -L/path 就很重要了！否则会找不到函式库喔！

除了连结的函式库之外，您或许已经发现一个奇怪的地方，那就是在我们的 sin.c 当中第一行『 #include <stdio.h>』，这行说的是要将一些定义数据由 stdio.h 这个档案读入，这包括 printf 的相关设定。这个档案其实是放置在 /usr/include/stdio.h 的！那么万一这个档案并非放置在这里呢？那么我们就可以使用底下的方式来定义出要读取的 include 档案放置的目录：

```
[root@linux ~]# gcc sin.c -lm -I/usr/include
```

-I/path 后面接的路径(Path)就是设定要去搜寻相关的 include 档案的目录啦！不过，同样的，默认值是放置在 /usr/include 底下，除非您的 include 档案放置在其它路径，否则也可以略过这个项目！

透过上面的几个小范例，您应该对于 gcc 以及原始码有一定程度的认识了，再接下来，我们来稍微整理一下 gcc 的简易使用方法吧！

---

gcc 的用法

前面说过， gcc 为 Linux 上面最标准的编译器，这个 gcc 是由 GNU 所维护的，有兴趣的朋友请自行前往参考。既然 gcc 对于 Linux 上的 Open source 是这么样的重要，所以底下我们就列举几个 gcc 常见的参数，如此一来大家应该更容易了解原始码的各项功能吧？！

```
# 仅将原始码编译成为目标文件，并不制作连结等功能：
[root@linux ~]# gcc -c hello.c
# 会自动的产生 hello.o 这个档案，但是并不会产生 binary 执行档。

# 在编译的时候，依据作业环境给予最佳化执行速度
```

```
[root@linux ~]# gcc -O hello.c -c
# 会自动的产生 hello.o 这个档案，并且进行最佳化喔！

# 在进行 binary file 制作时，将连结的函式库与相关的路径填入
[root@linux ~]# gcc sin.c -lm -L/usr/lib -I/usr/include
# 这个指令较常下达在最终连结成 binary file 的时候，
# -lm 指的是 libm.so 或 libm.a 这个函式库档案；
# -L 后面接的路径是刚刚上面那个函式库的搜寻目录；
# -I 后面接的是原始码内的 include 档案之所在目录。

# 将编译的结果输出成某个特定档名
[root@linux ~]# gcc -o hello hello.c
# -o 后面接的是要输出的 binary file 檔名

# 在编译的时候，输出较多的讯息说明
[root@linux ~]# gcc -o hello hello.c -Wall
# 加入 -Wall 之后，程序的编译会变的较为严谨一点，
# 所以警告讯息也会显示出来！
```

比较重要的大概就是这一些。另外，我们通常称 -Wall 或者 -O 这些非必要的参数为旗标(FLAGS)，因为我们使用的是 GCC ，所以有时候也会简称这些旗标为 CCFLAGS ，这些变量偶尔会被使用的喔！尤其是在后头会介绍的 make 相关的用法时，更是重要的很呐！ ^_^

---

make 的简易用法

在前言的部分我们提到过 make 的功能是可以简化编译过程里面所下达的指令， 同时还具有很多很方便的功能！那么底下咱们就来试看看使用 make 简化下达编译指令的流程吧！

为什么要用 make

先来想象一个案例，假设我的执行档里面包含了四个原始码档案，分别是 main.c haha.c sin_value.c cos_value.c 这四个档案，这四个档案您可以到 http://linux.vbird.org/linux_basic/0520source/main.tgz 来下载，由于这四个档案里面包含了相关性，并且还用到数学函式在里面， 所以如果您想要让这个程序可以跑，那么就需要这样编译：

```
[root@linux ~]# gcc -c main.c
[root@linux ~]# gcc -c haha.c
[root@linux ~]# gcc -c sin_value.c
[root@linux ~]# gcc -c cos_value.c
# 先以上面的动作制作出四个目标文件，然后再进行下面的动作：

[root@linux ~]# gcc -o main main.o haha.o sin_value.o cos_value.o \
> -lm -L/usr/lib -L/lib
# 这样就可以制作出  main 这个执行档啰！执行看看吧！
```

```
[root@linux ~]# ./main
HaHa! I'm the King of the world
0.706825
0.707388
```

呵呵！要做好多动作啊！真是麻烦，如果可以的话，能不能一个步骤就给他完成上面所有的动作呢？ 试看看在这个目录下建立一个 makefile 档案，内容如下：

```
# 1. 先建立编译的规则
[root@linux ~]# vi makefile
main: main.o haha.o sin_value.o cos_value.o
          gcc -o main main.o haha.o sin_value.o cos_value.o -lm
# 注意：第二行的 gcc 之前是 <tab> 按键产生的空格喔！

# 2. 尝试给他建立规则看看
[root@linux ~]# rm -f main *.o   <==先将之前的目标文件去除
[root@linux ~]# make
cc    -c -o main.o main.c
cc    -c -o haha.o haha.c
cc    -c -o sin_value.o sin_value.c
cc    -c -o cos_value.o cos_value.c
gcc -o main main.o haha.o sin_value.o cos_value.o -lm
# 这个时候 make 会主动去读取 makefile 这个档案的内容，
# 并根据内容直接去给他编译起相关的执行档啰！

# 3. 如果再执行一次 make 会怎样？！
[root@linux ~]# make
make: `main' is up to date.
# 看到了吧？！是否很方便呢？！
```

或许您会说：『如果我建立一个 shell script 来将上面的所有动作都集结在一起， 不是具有同样的效果吗？』呵呵！效果当然不一样，以上面的测试为例，我们仅写出 main 需要的目标文件，结果 make 会主动的去判断每个目标文件相关的原始码档案，并直接予以编译， 最后再直接进行连结的动作！哈哈！真的是很方便啊！此外，如果我们更动过某些原始码档案，则 make 也可以主动的判断哪一个原始码与相关的目标文件档案有更新过， 并仅更新该档案，如此一来，将可大大的节省很多编译的时间呢！要知道，某些程序在进行编译的行为时，会消耗很多的 CPU 资源呢！所以说， make 有这些好处：

- 简化编译时所需要下达的指令；
- 若在编译完成之后，修改了某个原始码档案，则 make 仅会针对被修改了的档案进行编译，其它的 object file 不会被更动；
- 最后可以依照相依性来更新(update)执行档。

既然 make 有这么多的优点，那么我们当然就得好好的了解一下 make 这个令人关心的家伙啦！而 make 里面最需要注意的大概就是那个规则档案，也就是 makefile 这个档案的语法啦！底下我们针对 makefile 的语法来加以介绍啰。

make 的基本语法与变量

make 的语法可是相当的多而复杂的，有兴趣的话可以到 http://www.gnu.org/software/make/manual/html_mono/make.html 去查阅相关的说明，鸟哥这里仅列出一些基本的规则，重点在于让读者们未来在接触原始码时，不会太紧张啊！ ^_^好了，基本的 makefile 规则是这样的：

```
标的(target): 目标文件1 目标文件2
<tab>   gcc -o 欲建立的执行文件 目标文件1 目标文件2
```

那个标的(target)就是我们想要建立的信息，而目标文件就是具有相关性的 object files ，那建立执行文件的语法就是以 <tab> 按键开头的那一行！特别给他留意喔， 『命令列必须要以 tab 按键作为开头』才行！ 他的规则基本上是这样的：

- 在 makefile 当中的 # 代表批注；
- <tab> 需要在命令行的第一个字符；
- 标的(target)与相依档案(就是目标文件)之间需以『:』隔开。

同样的，我们以刚刚上一个小节的范例进一步说明，如果我想要有两个以上的执行动作时， 例如下达一个指令就直接清除掉所有的目标文件与执行文件，该如何制作呢？

```
# 1. 先建立编译的规则
[root@linux ~]# vi makefile
main: main.o haha.o sin_value.o cos_value.o
          gcc -o main main.o haha.o sin_value.o cos_value.o -lm
clean:
          rm -f main main.o haha.o sin_value.o cos_value.o

# 2. 测试看看：
[root@linux ~]# make clean
rm -rf main main.o haha.o sin_value.o cos_value.o
```

如此一来，我们的 makefile 里面就具有至少两个标的，分别是 main 与 clean ，如果我们想要建立 main 的话，输入『make main』，如果想要清除有的没的，输入『make clean』即可啊！而如果想要先清除目标文件再编译 main 这个程序的话，就可以这样输入：『make clean main』，如下所示：

```
[root@linux ~]# make clean main
rm -rf main main.o haha.o sin_value.o cos_value.o
cc    -c -o main.o main.c
cc    -c -o haha.o haha.c
cc    -c -o sin_value.o sin_value.c
cc    -c -o cos_value.o cos_value.c
gcc -o main main.o haha.o sin_value.o cos_value.o -lm
```

这样就很清楚了吧！？但是，您是否会觉得，咦！ makefile 里面怎么重复的数据这么多啊！呵呵！没错！所以我们可以再藉由 shell script 那时学到的『变数』来更简化 makefile 喔：

```
[root@linux ~]# vi makefile
LIBS = -lm
OBJS = main.o haha.o sin_value.o cos_value.o
main: ${OBJS}
        gcc -o main ${OBJS} ${LIBS}
clean:
        rm -f main ${OBJS}
```

与 bash shell script 的语法有点不太相同，变量的基本语法为：

1. 变量与变量内容以『=』隔开，同时两边可以具有空格；
2. 变量左边不可以有 <tab> ，例如上面范例的第一行 LIBS 左边不可以是 <tab>；
3. 变量与变量内容在『=』两边不能具有『:』；
4. 在习惯上，变数最好是以『大写字母』为主；
5. 运用变量时，以 ${变量} 或 $(变量) 使用；
6. 在该 shell 的环境变量是可以被套用的，例如提到的 CFLAGS 这个变数！
7. 在指令列模式也可以给予变量。

由于 gcc 在进行编译的行为时，会主动的去读取 CFLAGS 这个环境变量，所以，您可以直接在 shell 定义出这个环境变量，也可以在 makefile 档案里面去定义，更可以在指令列当中给予这个咚咚呢！例如：

```
[root@linux ~]# CFLAGS="-Wall" make clean main
# 这个动作在上 make 进行编译时，会去取用 CFLAGS 的变量内容！
```

也可以这样：

```
[root@linux ~]# vi makefile
LIBS = -lm
OBJS = main.o haha.o sin_value.o cos_value.o
CFLAGS = -Wall
main: ${OBJS}
          gcc -o main ${OBJS} ${LIBS}
clean:
          rm -f main ${OBJS}
```

咦！我可以利用指令列进行环境变量的输入，也可以在档案内直接指定环境变量，那万一这个 CFLAGS 的内容在指令列与 makefile 里面并不相同时，以那个方式输入的为主？呵呵！ 环境变量取用的规则是这样的：

1. make 指令列后面加上的环境变量为优先；
2. makefile 里面指定的环境变量第二；
3. shell 原本具有的环境变量第三。

此外，还有一些特殊的变量需要了解的喔：

- $@：代表目前的标的(target)

所以我也可以将 makefile 改成：

```
[root@linux ~]# vi makefile
LIBS = -lm
OBJS = main.o haha.o sin_value.o cos_value.o
CFLAGS = -Wall
main: ${OBJS}
        gcc -o $@ ${OBJS} ${LIBS}   <==那个 $@ 就是 main !
clean:
        rm -f main ${OBJS}
```

这样是否稍微了解了 makefile (也可能是 Makefile) 的基本语法？这对于您未来自行修改原始码的编译规则时，是很有帮助的喔！^_^！

Tarball 的管理与建议：

好了！在我们知道了原始码的相关信息之后，再来要了解的自然就是如何使用具有原始码的 Tarball 来建立一个属于自己的套件啰！从前面几个小节的说明当中，我们晓得其实 Tarball 的安装是可以跨平台的，因为 C 语言的程序代码在各个平台上面是可以共通的，只是需要的编译器可能并不相同而已。例如 Linux 上面用 gcc 而 Windows 上面也有相关的 C 编译器啊～所以呢，同样的一组原始码，既可以在 Fedora Linux 上面编译，也可以在 SuSE Linux 上面编译，当然，也可以在大部分的 Unix 平台上面编译成功的！

所以啰， Tarball 原始码程序应该可以在大部分的环境底下安装成功的！举例来说，鸟哥在上面几个小节所提供的 C 程序是在 Fedora Core 4 及 Red Hat 9 上面测试编译的，那么您可以下载之后在自己的 Linux 环境下测试看看，我想，每个人应该都可以顺利的编译成功的才是！因为 C 的语法是没有不一样的啊！ ^_^

如果万一没有编译成功怎么办？很简单啊，透过修改小部分的程序代码(通常是因为很小部分的异动而已) 就可以进行跨平台的移植了！也就是说，刚刚我们在 Linux 底下写的程序『理论上，是可以在 Windows 上面编译的！』这就是原始码的好处啦！所以说，如果朋友们想要学习程序语言的话， 鸟哥个人是比较建议学习『具有跨平台能力的程序语言』，例如 C 就是很不错的一个！

唉啊！又扯远了～赶紧拉回来继续说明我们的 Tarball 啦！

使用原始码管理套件所需要的基础套件

从原始码的说明我们晓得要制作一个 binary 执行档需要很多咚咚的呢！这包括底下这些基础的套件：

- gcc 或 cc 等 C 语言编译器(compiler)：
  这是一定要的啦！要将原始码编译成为可执行的 binary 才行，所以当然就需要编译器啰！在 Linux 上面用的当然就是 GNU 发展的 gcc 这个超好用的免费的 C 编译器啦！并且，很多在 Linux 平台上面发展的套件的原始码，原本就是以 gcc 为底来设计的呢。
- make 及 autoconfig 等套件：
  一般来说，以 Tarball 方式释出的套件当中，为了简化编译的行程，通常都是配合前几个小节提到的 make 这个指令来依据目标档案的相依性而进行编译。但是我们也知道说 make 需要 makefile 这个档案的规则，那由于不同的系统里面可能具有的基础套件环境并不相同， 所以就

需要侦测使用者的作业环境，好自行建立一个 makefile 档案。这个自行侦测的小程序也必须要藉由 autoconfig 这个相关的套件来辅助才行。

- 需要 Kernel 提供的 Library 以及相关的 Include 档案：
  从前面的原始码编译过程，我们晓得函式库(library)的重要性，同时也晓得有 include 档案的存在。很多的套件在发展的时候都是直接取用系统核心提供的函式库与 include 档案的，这样才可以与这个操作系统兼容啊！尤其是在『 驱动程序方面的套件 』，例如网络卡、声卡、USB 等驱动程序在安装的时候，常常是需要核心提供的相关信息的。在 Red Hat 的系统当中 (包含 Fedora 等系列) ，这个核心相关的功能通常都是被包含在 kernel-source 或 kernel-header 这些套件名称当中，所以记得要安装这些套件喔！

虽然 Tarball 的安装上面相当的简单，如同我们前面几个小节的例子，只要顺着开发商提供的 README 与 INSTALL 档案所载明的步骤来进行，安装是很容易的。但是我们却还是常常会在 BBS 或者是新闻群组当中发现这些留言：『我在执行某个程序的侦测档案时，他都会告诉我没有 gcc 这个套件，这是怎么回事？』还有：『我没有办法使用 make 耶！这是什么问题？』呵呵！ 这就是没有安装上面提到的那些基础套件啦！

咦！为什么使用者不安装这些套件啊？呵呵！这是因为目前的 Linux distribution 大多已经偏向于桌上型计算机的使用，他们希望使用者能够按照厂商自己的希望来安装相关的套件即可， 所以通常『预设』是没有安装 gcc 或者是 make 等套件的。所以啦，如果您希望未来可以自行安装一些以 Tarball 方式释出的套件时，记得请自行挑选想要安装的套件名称喔！例如在 Fedora 或者是 Red Hat 当中记得选择 Software Development 以及 Kernel Source Development 等相关字眼的群集呢。

那万一我已经安装好一部 Linux 主机，但是使用的是默认值所安装的套件，所以没有 make, gcc 等咚咚，该如何是好？呵呵！问题其实不大啦，目前使用最广泛的 Fedora 或者是 Red Hat 大多是以 RPM (下一章会介绍) 来安装套件的，所以，您只要拿出当初安装 Linux 时的原版光盘，然后以下一章介绍的 RPM 来一个一个的加入到您的 Linux 主机里面就好啦！很简单的啦！

Tarball 安装的基本步骤

我们提过以 Tarball 方式释出的套件是需要重新编译可执行的 binary file 的。而 Tarball 是以 tar 这个指令来打包与压缩的档案，所以啦，当然就需要先将 Tarball 解压缩，然后到原始码所在的目录下进行 makefile 的建立，再以 make 来进行编译与安装的动作啊！所以整个安装的基础动作大多是这样的：

1. 将 tarball 档案在 /usr/local/src 目录下解压缩；
2. 进入新建立的目录底下，去查阅 INSTALL 与 README 等相关档案内容(很重要的步骤！)；
3. 根据 INSTALL/README 的内容察看并安装好一些相依的套件(非必要)；
4. 以自动侦测程序(configure 或 config)侦测作业环境，并建立 Makefile 这个档案；
5. 以 make 这个程序并使用该目录下的 Makefile 做为他的参数设定档，来进行 make (编译或其它)的动作；
6. 以 make 这个程序，并以 Makefile 这个参数设定档，依据 install 这个标的(target)的指定来安装到正确的路径！

注意到上面的第二个步骤，通常在每个软件在释出的时候，都会附上 INSTALL 或者是 README 这种档名的说明档，这些说明档请『确实详细的』 阅读过一遍，通常这些档案会记录这个软件的安装要求、软件的工作项目、 与软件的安装参数设定及技巧等，只要仔细的读完这些档案，基本上，要安装好 tarball 的档

案，都不会有什么大问题啰。至于 makefile 在制作出来之后，里头会有相当多的标的(target)，最常见的就是 install 与 clean 啰！通常『make clean』代表着将目标文件(object file)清除掉，『make』则是将原始码进行编译而已。注意喔!编译完成的可执行档与相关的设定档还在原始码所在的目录当中喔!因此，最后要进行『make install』来将编译完成的所有咚咚都给他安装到正确的路径去，这样就可以使用该套件啦！

OK！我们底下约略提一下大部分的 tarball 软件之安装的指令下达方式：

1. ./configure
   这个步骤就是在建立 Makefile 这的档案啰！通常程序开发者会写一支 scripts 来检查您的 Linux 系统、相关的套件属性等等，这个步骤相当的重要， 因为未来您的安装信息都是这一步骤内完成的！另外，这个步骤的相关信息应该要参考一下该目录下的 README 或 INSTALL 相关的档案！！基本上，这个步骤完成之后会建立(或修改)一个 Makefile ，这就是参数档啦！
2. make clean
   make 会读取 Makefile 中关于 clean 的工作。这个步骤不一定会有，但是希望执行一下！ 为什么呢？因为在进行编译的时候，会产生一些 *.o 的档案，例如有个 abc.c 的原始码，经过编译后会变成 abc.o 的档案！我们称这些档案为 object file ，这些档案如果之前已经编译过并留下来的话，那么这次再编译的时候， 就不会编译该档案，然而由于我们可能已经修改了部分的参数，因此该档案的编译结果事实 上应该会有所不同！因此，为了避免前一次留下来的数据可能影响到这次编译的结果， 所以通常可以进行一下这个步骤啰！
3. make
   make 会依据 Makefile 当中的预设工作进行编译的行为！编译的工作主要是进行 gcc 来将原始码编译成为可以被执行的 object files ，但是这些 object files 通常还需要一些函式库之类的 link 后，才能产生一个完整的执行档！使用 make 就是要将原始码编译成为可以被执行的可执行档，而这个可执行档会放置在目前所在的目录之下， 尚未被安装到预定安装的目录中；
4. make install
   通常这就是最后的安装步骤了，make 会依据 Makefile 这个档案里面关于 install 的项目，将上一个步骤所编译完成的数据给他安装到预定的目录中， 就完成安装啦！

请注意，上面的步骤是一步一步来进行的，而 其中只要一个步骤无法成功，那么后续的步骤就完全没有办法进行的！ 因此，要确定每一的步骤都是成功的才可以！举个 例子来说，万一今天你在 ./configure 就不成功了，那么就表示 Makefile 无法被建立起来，要知道，后面的步骤都是根据 Makefile 来进行的，既然无法建立 Makefile ，后续的步骤当然无法成功啰！另外，如果在 make 无法成功的话，那就表示源文件无法被编译成可执行档，那么 make install 主要是将编译完成的档案给他安装下去的，既然都没有成功的执行档了，怎么进行安装？ 所以啰，要每一个步骤都正确无误才能往下继续做！此外，如果安装成功， 并且是安装在独立的一个目录中，例如 /usr/local/packages 这个目录中好了，那么您就必需手动的将这个套件的 man page 给他放到 /etc/man.config 里面去。

一般 Tarball 套件安装的建议事项(如何移除？升级？)

或许您已经发现了也说不定，那就是为什么前一个小节里面， Tarball 要在 /usr/local/src 里面解压缩呢？呵呵！基本上，在预设的情况下，原本的 Linux distribution 释出安装的套件大多是在 /usr 里面的，而使用者自行安装的套件则建议放置在 /usr/local 里面。这是考虑到管理使用者所安装套件的便利性。

怎么说呢？我们晓得几乎每个套件都会提供在线说明的服务，那就是 info 与 man 的功能。在预设的情况下， man 会去搜寻 /usr/local/man 里面的说明文件， 因此，如果我们将套件安装在 /usr/local 底下的话，那么自然安装完成之后， 该套件的说明文件就可以被找到了。此外，如果您所管理的主机其实是由多人共同管理的， 或者是如同学校里面，一部主机是由学生管理的，但是学生总会毕业吧？ 所以需要进行交接，如果大家都将套件安装在 /usr/local 底下，那么管理上不就显的特别的容易吗？！

所以啰，通常我们会建议大家将自己安装的套件放置在 /usr/local 下，至于原始码(Tarball)则建议放置在 /usr/local/src(src 为 source 的缩写)底下啊。

再来，让我们先来看一看 Linux distribution 预设的安装套件的路径会用到哪些？我们以 apache 这个软件来说明的话(apache 是 WWW 服务器软件，详细的数据请参考服务器架设篇。 您的系统不见得有装这个套件)：

- /etc/httpd
- /usr/lib
- /usr/bin
- /usr/share/man

我们会发现套件的内容大致上是摆在 etc, lib, bin, man 等目录当中，分别代表『设定档、函式库、执行档、在线说明档』。 好了，那么你是以 tarball 来安装时呢？如果是放在预设的 /usr/local 里面，由于 /usr/local 原本就预设这几个目录了，所以你的数据就会被放在：

- /usr/local/etc
- /usr/local/bin
- /usr/local/lib
- /usr/local/man

但是如果你每个套件都选择在这个预设的路径下安装的话， 那么所有的套件的档案都将放置在这四个目录当中，因此，如果你都安装在这个目录下的话， 那么未来再想要升级或移除的时候，就会比较难以追查档案的来源啰！ 而如果您在安装的时候选择的是单独的目录，例如我将 apache 安装在 /usr/local/apache 当中，那么您的档案目录就会变成：

- /usr/local/apache/etc
- /usr/local/apache/bin
- /usr/local/apache/lib
- /usr/local/apache/man

呵呵呵呵！单一套件的档案都在同一个目录之下，那么要移除该套件就简单的多了！ 只要将该目录移除即可视为该套件已经被移除啰 ！以上面为例，我想要移除 apache 只要下达『rm -rf /usr/local/apache』就算移除这个套件啦！当然啰，实际安装的时候还是得视该软件的 Makefile 里头的 install 信息才能知道到底他的安装情况为何的。因为例如 sendmail 的安装就很麻烦......这个方式虽然有利于套件的移除，但不晓得您有没有发现， 我们在执行某些指令的时候，与该指令是否在 PATH 这个环境变量所记录的路径有关，以上面为例，我的 /usr/local/apache/bin 肯定是不在 PATH 里面的，所以执行 apache 的指令就得要利用绝对路径了，否则就得将这个 /usr/local/apache/bin 加入 PATH 里面。另外，那个

/usr/local/apache/man 也需要加入 man page 搜寻的路径当中啊！

除此之外， Tarball 在升级的时候也是挺困扰的，怎么说呢？我们还是以 apache 来说明好了。WWW 服务器为了考虑互动性，所以通常会将 PHP+MySQL+Apache 一起安装起来(详细的信息请参考服务器架设篇)，果真如此的话，那么每个套件在安装的时候『 都有一定的顺序与程序！』因为他们三者之间具有相关性，所以安装时必需要三者同时考虑到他们的函式库与相关的编译参数。那么如果今天我只要升级 PHP 呢？有的时候因为只有涉及动态函式库的升级，那么我只要升级 PHP 即可！其它的部分或许影响不大。但是如果今天 PHP 需要重新编译的模块比较多，那么可能会连带的，连 Apache 这个程序也需要重新编译过才行！真是有点给他头痛的！没办法啦！使用 tarball 确实有他的优点啦，但是在这方面，确实也有他一定的伤脑筋程度。

由于 Tarball 在升级与安装上面具有这些特色，亦即 Tarball 在反安装上面具有比较高的难度(如果您没有好好规划的话～)，所以，为了方便 Tarball 的管理，通常我们会这样建议使用者：

1. 最好将 tarball 的原始数据解压缩到 /usr/local/src 当中；
2. 安装时，最好安装到 /usr/local 这个预设路径下；
3. 考虑未来的反安装步骤，最好可以将每个套件单独的安装在 /usr/local 底下：
   例如安装 rp-pppoe-2.6.tar.gz 时，则可以指定该套件需要安装于 /usr/local/rp-pppoe 当中，如此一来，该套件会将所有的数据都写入 /usr/local/rp-pppoe 当中，因此，未来如果要移除该套件，只要将该目录删除即可视为成功的移除了！
4. 加上 man path
   不过单独安装某个套件在某一特定路径下的作法，会导致当有 man page 的时候，使用预设的 MANPATH 会找不到相关的说明档案内容。这个时候就必须要将 man page 的路径加到 /etc/man.config 档案中了！否则使用 man 也查询不到指令的使用方法的。以上面的例子为例，如果是安装了 /usr/local/rp-pppoe 当中，通常 man page 会放在 /usr/local/rp-pppoe/man 当中，所以，您就必需要在 /etc/man.config 里面差不多 40~50 行左右的地方，加入底下这一行：

   ```
   MANPATH /usr/local/rp-pppoe/man
   ```

   这样就可以使用 man 来查询资料啰！

一个简单的范例、利用 ntp 来示范

读万卷书不如行万里路啊！所以当然我们就来给他测试看看，看您是否真的了解了如何利用 Tarball 来安装软件呢？！我们利用时间服务器 ntp-4.1.2 这个套件来测试安装看看。先请到 http://www.ntp.org/downloads.html 这个目录去下载档案，(您也可以下载比较新的档案来测试的啦！) 或者直接到鸟哥的网站下载：

http://linux.vbird.org/linux_basic/0520source/ntp-stable-4.2.0a-20050816.tar.gz。

假设我对这个套件的要求是这样的：

- 假设 ntp-stable-4.2.0a-20050816.tar.gz 这个档案放置在 /root 这个目录下；

* 原始码请解开在 /usr/local/src 底下；
* 我要安装到 /usr/local/ntp 这个目录中；

那么您可以依照底下的步骤来安装测试看看(如果可以的话，请您不要参考底下的文件数据， 先自行安装过一遍这个软件，然后再来对照一下鸟哥的步骤喔！)。

```
# 1. 解压缩，并阅读一下 ntp 底下的 README 与 INSTALL：
[root@linux ~]# cd /usr/local/src
[root@linux src]# tar -zxvf /root/ntp-stable-4.2.0a-20050816.tar.gz
# 这个步骤会让原始码解开成为 /usr/local/src/ntp-stable-4.2.0a-20050816 这个目录

# 2. 进入原始码所在目录，并且查阅如何安装的技巧：
[root@linux src]# cd ntp-stable-4.2.0a-20050816
[root@linux ntp*]# vi INSTALL ( 或 vi README )

# 3. 开始设定参数、编译与安装：
[root@linux ntp*]# ./configure --help | more
# 上面这个动作可以察看一下可用的参数！

[root@linux ntp*]# ./configure --prefix=/usr/local/ntp \
>  --enable-all-clocks --enable-parse-clocks
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
......中间省略......
config.status: creating util/Makefile
config.status: creating config.h
config.status: executing depfiles commands
# 一般来说 configure 设定参数较重要的就是那个 --prefix=/path 了，
# --prefix 后面接的路径就是『这个软件未来要安装到那个目录去？』
# 如果您没有指定 --prefix=/path 这个参数，通常预设参数就是 /usr/local
# 至于其它的参数意义就得要参考 ./configure --help 了！
# 这个动作完成之后会产生 makefile 或 Makefile 这个档案
# 当然啦，这个侦测检查的过程会显示在屏幕上，特别留意关于 gcc 的检查，
# 还有最重要的是最后需要成功的建立起 Makefile 才行！(上面最后一行)

# 4. 编译与安装：
[root@linux ntp*]# make clean; make
[root@linux ntp*]# make check
[root@linux ntp*]# make install
# 将数据给他安装在 /usr/local/ntp 底下
```

整个动作就这么简单，您完成了吗？！完成之后到 /usr/local/ntp 您发现了什么？！

利用 patch 更新原始码

我们在前言里面介绍了为何需要进行套件的升级，这是很重要的喔！那假如我是以 Tarball 来进行某个套件的安装，那么是否当我要升级这个套件时，就得要下载这个套件的完整全新的 Tarball 呢？举个例子来说，鸟哥有个讨论区在 http://phorum.vbird.org 这个网址，这个讨论区是以 phpBB 这个套件来架设的，而鸟哥的讨论区版本为 phpbb2.0.1.tar.gz ，目前(2005/09/30)最新释出的版本则是 phpbb2.0.17.tar.gz 。那我是否需要下载全新的 phpbb2.0.17.tar.gz 这个档案来更新原本的旧程序呢？

事实上，当我们发现一些套件的漏洞，通常是某一段程序代码写的不好所致。因此， 所谓的『更新原始码』常常是只有更改部分档案的小部分内容而已。 既然如此的话，那么我们是否可以就那些被更动的档案来进行修改就可以咯？ 也就是说，旧版本到新版本间没有更动过的档案就不要理他，仅将有修订过的档案部分来处理即可。 这有什么好处呢？首先，没有更动过的档案的目标文件(object file)根本就不需要重新编译，而且有更动过的档案又可以利用 make 来自动 update （ 更新 ），如此一来，呵呵！我们原先的设定(makefile 档案里面的规则)将不需要重新改写或侦测！呵呵！可以节省很多宝贵的时间呢(例如后续章节会提到的核心的编译！)

从上面的说明当中，我们可以发现，如果可以将旧版的原始码数据改写成新版的版本， 那么就能直接编译了，而不需要将全部的新版 Tarball 重新下载一次呢！可以节省频宽与时间说！那么如何改写原始码？ 难道要我们一个档案一个档案去参考然后修订吗？当然没有这么没人性！

我们在正规表示法的时候有提到一个比对两个档案的指令，那就是 diff ，这个指令可以将『两个档案之间的差异性列出来』呢！ 那我们也知道新旧版本的档案之间，其实只有修改一些程序而已，那么我们可以透过 diff 比对出新旧版本之间的文字差异，然后再以相关的指令来将旧版的档案更新吗？！ 呵呵！当然可以啦！那就是 patch 这个指令啦！很多的套件开发商在更新了原始码之后，几乎都会释出所谓的 patch file，也就是直接将原始码 update 而已的一个方式喔！我们底下以一个简单的范例来说明给您了解喔！

假设我们有两个档案，分别是 expatch.old 与 expatch.new ，他们的内容是这样的：

```
[root@linux ~]# vi expatch.old
echo "check your postfix's body and header drop settings"
echo "postmap -q - regexp:header_checks < header_checks"
postmap -q - regexp:header_checks < header_checks
echo "postmap -q - regexp:body_checks < body_checks"
postmap -q - regexp:body_checks < body_checks

[root@linux ~]# vi expatch.new
echo "check your postfix's body and header drop settings"
echo "postmap -q - regexp:header_checks < header_checks This's right"
postmap -q - regexp:header_checks < header_checks
echo "postmap -q - regexp:body_checks < body_checks This's right"
postmap -q - regexp:body_checks < body_checks
```

两个档案的不同点在于：

```
[root@linux ~]# diff expatch.old expatch.new
2c2
```

```
< echo "postmap -q - regexp:header_checks < header_checks"
---
> echo "postmap -q - regexp:header_checks < header_checks This's right"
4c4
< echo "postmap -q - regexp:body_checks < body_checks"
---
> echo "postmap -q - regexp:body_checks < body_checks This's right"
```

上面显示出两个档案的不同点，详细的意义请参考正规表示法那个章节的介绍。 好了，假如我以『 diff -c expatch.old expatch.new 』以及上面显示的信息，做成一个档案，内容是这样的：

```
[root@linux ~]# diff -Naur expatch.old expatch.new > expatch.patch
[root@linux ~]# vi expatch.patch
--- expatch.old 2005-09-30 15:47:54.000000000 +0800
+++ expatch.new 2005-09-30 15:48:06.000000000 +0800
@@ -1,5 +1,5 @@
 echo "check your postfix's body and header drop settings"
-echo "postmap -q - regexp:header_checks < header_checks"
+echo "postmap -q - regexp:header_checks < header_checks This's right"
 postmap -q - regexp:header_checks < header_checks
-echo "postmap -q - regexp:body_checks < body_checks"
+echo "postmap -q - regexp:body_checks < body_checks This's right"
 postmap -q - regexp:body_checks < body_checks
```

注意到，这个档案的第一行显示出旧版本的文件名，而第二行则为新版本的档名与时间， 第三行以后则是两个档案的差异性。那么我们将以 patch 来进行更新，将 expatch.old 更新到 expatch.new 看看。patch 的基本语法是这样的：

patch -p 数字 < patch_file

特别留意那个 -p 数字，那是与 patch_file 里面列出的文件名有关的信息。假如在 patch_file 第一行写的是这样：

*** /home/guest/example/expatch.old

那么当我下达『 patch -p0 < patch_file 』时，则更新的档案是『 /home/guest/example/expatch.old 』，如果『 patch -p1 < patch_file』，则更新的档案为『home/guest/example/expatch.old』，如果『patch -p4 < patch_file』则更新『expatch.old』，也就是说， -pxx 那个 xx 代表『拿掉几个斜线(/)』的意思！这样可以理解了吗？！好了，那么我要开始来更新我的 expatch.old 了，可以这样搞定：

```
[root@linux ~]# patch -p0 < expatch.patch
patching file expatch.old
# 注意喔，这个时候我的工作目录底下会存在 expatch.old 才对！
# 然后立刻察看一下，您会发觉， expatch.new 与 expatch.old 变成一模一样的了！
```

很容易了解吧！上面三个档案您可以在底下的网址取得：

http://linux.vbird.org/linux_basic/0520source/expatch.tgz

加油的啦！另外，如果您是以 patch 更新原始码，那么记得， 您可能必须要重新编译，并且重新 install 才算成功更新喔！并不是 patch 就好了！因为 patch 的功能主要仅只是更新原始码档案而已！切记切记！

鸟哥提问题：如果我有一个很旧版的套件，这个套件已经更新到很新的版本，例如核心，那么我可以使用 patch file 来更新吗？

这个问题挺有趣的，首先，您必须要确定旧版本与新版本之间『确实有释出 patch file 』才行，以 kernel 2.2.xx 及 2.4.xx 来说，这两者基本上的架构已经不同了，所以两者间是无法以 patch file 来更新的。不过， 2.4.xx 与 2.4.yy 就可以更新了。不过，因为 kernel 每次推出的 patch 档案都仅针对前一个版本而已，所以假设要由 kernel 2.4.20 升级到 2.4.26 ，就必须要使用 patch 2.4.21, 2.4.22, 2.4.23, 2.4.24, 2.4.25, 2.4.26 六个档案来『依序更新』才行喔！当然，如果有朋友帮您比对过 2.4.20 与 2.4.26 ，那您自然就可以使用该 patch file 来直接一次更新啰！

函式库管理：

在我们的 Linux 操作系统当中，函式库是很重要的一个项目。 因为很多的套件之间都会互相取用彼此提供的函式库来进行特殊功能的运作，例如很多需要验证身份的程序都习惯利用 PAM 这个模块提供的验证机制来实作，而很多网络联机机制则习惯利用 SSL 函式库来进行联机加密的机制。所以说，函式库的利用是很重要的。不过， 函式库又依照是否被编译到程序内部而分为动态与静态函式库，这两者之间有何差异？哪一种函式库比较好？ 底下我们就来谈一谈先！

动态与静态函式库

首先我们要知道的是，函式库的类型有哪些？依据函式库被使用的类型而分为两大类，分别是静态(Static) 与动态(Dynamic)函式库两类。底下我们来谈一谈这两种类行的函式库吧！

- 静态函式库：
  - 附檔名：
    这类的函式库通常附档名为 libxxx.a 的类型；
  - 编译行为：
    这一类型的函式库在被使用到程序当中的时候，都是整个函式库的所有数据被整合到执行文件当中， 也就是说，当我们在进行编译的动作时，这个函式库会被加入到执行档内，所以 利用静态函式库编译成的档案会比较大一些喔；
  - 独立执行的状态：
    这类函式库最大的优点，就是编译成功的可执行档 可以独立执行，而不需要再向外部要求读取函式库的内容(请参照动态函式库的说明)。
  - 升级难易度：
    虽然执行档可以独立执行，然而当函式库升级的时候， 由于我们的执行档取用的是之前版本的函式库，所以当函式库升级后， 连执行档也需要重新编译过一次，才能将新的函式库整合到执行档当中。

- 动态函式库：
  - 附檔名：
    这类函式库通常附档名为 libxxx.so 的类型；

- 编译行为：
  动态函式库与静态函式库的编译行为差异挺大的， 静态函式库是整个被编译到执行文件当中，但是动态函式库在编译的时候，在程序里面只有一个『 指向(Pointer)』 的位置而已。 也就是说，动态函式库的内容并没有被整合到执行档当中，而是当执行档要使用到函式库的机制时， 程序才会去读取函式库来使用。由于执行文件当中仅具有指向动态函式库所在的指标而已， 并不包含函式库的内容，所以他的档案会比较小一点。
- 独立执行的状态：
  这类型的函式库不能被独立执行， 因为当我们使用到函式库的机制时，程序会去读取函式库，所以函式库『 必须要存在』才行，而且，函式库的『 所在目录也不能改变』， 因为我们的可执行档里面仅有『指标』亦即当要取用该动态函式库时，程序会主动去某个路径下读取， 呵呵！所以动态函式库可不能随意移动或删除，会影响很多相依的程序软件喔！
- 升级难易度：
  虽然这类型的执行档无法独立运作，然而由于是具有指向的功能， 所以，当函式库升级后，执行档根本不需要进行重新编译的行为，因为执行档会直接指向新的函式库档案(前提是函式库新旧版本的档名相同喔！)。

在目前的 Linux distribution 当中，我们比较倾向于使用动态函式库， 因为如同上面提到的最重要的一点，就是函式库的升级方便！由于 Linux 系统里面的套件相依性太复杂了，如果使用太多的静态函式库，那么升级某一个函式库时，都会对整个系统造成很大的冲击!因为其它相依的执行档也要同时重新编译啊!这个时候动态函式库可就有用多了，因为只要动态函式库升级就好，其它的套件根本无须变动。

那么这些函式库放置在哪里呢？绝大多数的函式库都放置在：

- /usr/lib
- /lib

此外，Linux 系统里面很多的函式库其实 kernel 就提供了，那么 kernel 的函式库放在哪里？呵呵！就是在 /lib/modules 里面啦！里面的数据可多着呢！不过要注意的是， 不同版本的核心提供的函式库差异性是挺大的，所以 kernel 2.4.xx 版本的系统不要想将核心换成 2.6.xx 喔！ 很容易由于函式库的不同而导致很多原本可以执行的软件套件无法顺利运作呢！ 更多的核心相关说明我们在后面会继续的给他介绍的。

---

ldconfig 与 /etc/ld.so.conf

在了解了动态与静态函式库，也知道我们目前的 Linux 大多是将函式库做成动态函式库之后，再来要知道的就是，那有没有办法增加函式库的读取效能？！ 我们知道内存的存取速度是硬盘的好几倍，所以， 如果我们将常用到的动态函式库先加载内存当中(快取，cache)，如此一来，当软件套件要取用动态函式库时，就不需要重从头由硬盘里面读出啰！ 这样不就可以增进动态函式库的读取速度？没错，是这样的！这个时候就需要 ldconfig 与 /etc/ld.so.conf 的协助了。

如何将动态函式库加载高速缓存(cache)当中呢？

1. 首先，我们必须要在 /etc/ld.so.conf 里面写下『 想要读入高速缓存当中的动态函式库所在的目录』，注意喔， 是目录而不是档案；

2. 接下来则是利用 ldconfig 这个执行档将 /etc/ld.so.conf 的资料读入快取当中；
3. 同时也将数据记录一份在 /etc/ld.so.cache 这个档案当中呐！

事实上， ldconfig 还可以用来判断动态函式库的连结信息呢！赶紧利用 Fedora Core 4 来测试看看。假设我还想要将我的 MySQL 函式库加入到快取当中：

```
[root@linux ~]# ldconfig [-f conf] [ -C cache] [-p]
参数：
-f conf ：那个 conf 指的是某个文件名称，也就是说，使用 conf 作为 libarary
            函式库的取得路径，而不以 /etc/ld.so.conf 为默认值
-C cache：那个 cache 指的是某个文件名称，也就是说，使用 cache 作为快取暂存
            的函式库资料，而不以 /etc/ld.so.cache 为默认值
-p      ：列出目前有的所有函式库资料内容（在 /etc/ld.so.cache 内的资料！）
范例：

范例一：假设我的 MySQL 数据库函式库在 /usr/lib/mysql 当中，如何读入 cache ？
[root@linux ~]# vi /etc/ld.so.conf
include ld.so.conf.d/*.conf
/usr/lib/mysql   <==这一行新增的啦！

[root@linux ~]# ldconfig
# 画面上不会显示任何的信息，不要太紧张！正常的！

[root@linux ~]# ldconfig -p
928 libs found in cache `/etc/ld.so.cache'
        libz.so.1 (libc6) => /usr/lib/libz.so.1
        libz.so (libc6) => /usr/lib/libz.so
......中间省略......
```

透过上面的动作，我们可以将 MySQL 的相关函式库给他读入快取当中， 这样可以加快函式库读取的效率呢！在某些时候，您可能会自行加入某些 Tarball 安装的动态函式库，而您想要让这些动态函式库的相关连结可以被读入到快取当中， 这个时候您可以将动态函式库所在的目录名称写入 /etc/ld.so.conf 当中，然后执行 ldconfig 就可以啦！

---

ldd

说了这么多，那么我如何判断某个可执行的 binary 档案含有什么动态函式库呢？很简单，利用 ldd 就可以晓得了！例如我想要知道 /usr/bin/passwd 这个程序含有的动态函式库有哪些，可以这样做：

```
[root@linux ~]# ldd [-vdr] [filename]
参数：
-v ：列出所有内容信息；
-d ：重新将资料有遗失的 link 点秀出来！
-r ：将 ELF 有关的错误内容秀出来！
范例：
```

```
范例一：找出 /usr/bin/passwd 这个档案的函式库数据
[root@linux ~]# ldd /usr/bin/passwd
        linux-gate.so.1 =>  (0x00d19000)
          .......中间省略......
        libpam_misc.so.0 => /lib/libpam_misc.so.0 (0x00bd6000)
          .......中间省略......
# 我们前言的部分不是一直提到 passwd 有使用到 pam 的模块吗？！怎么知道？
# 利用 ldd 察看一下这个档案，看到 libpam.so 了吧？这就是 pam 提供的函式库

范例二：找出 /lib/libc.so.6 这个函式的相关其它函式库！
[root@linux ~]# ldd /lib/libc.so.6
        /lib/ld-linux.so.2 (0x00bf1000)
        linux-gate.so.1 =>  (0x00632000)

[root@linux ~]# ldd -v /lib/libc.so.6
        /lib/ld-linux.so.2 (0x00bf1000)
        linux-gate.so.1 =>  (0x00111000)

        Version information:
        /lib/libc.so.6:
                ld-linux.so.2 (GLIBC_2.1) => /lib/ld-linux.so.2
                ld-linux.so.2 (GLIBC_2.3) => /lib/ld-linux.so.2
                ld-linux.so.2 (GLIBC_PRIVATE) => /lib/ld-linux.so.2
```

未来如果您常常升级安装 RPM 的套件时(下一章节会介绍)，应该常常会发现那个『 相依属性』的问题吧！？没错！我们可以先以 ldd 来视察『相依函式库』之间的相关性！以先取得了解！ 例如上面的例子中，我们检查了 libc.so.6 这个在 /lib 当中的函式库，结果发现他其实还跟 ld-linux.so.2 有关！所以我们就需要来了解一下，那个档案到底是什么套件的函式库呀！？使用 -v 这个参数还可以得知该函式库来自于哪一个套件！像上面的数据中，就可以得到该 libc.so.6 其实可以支持 GLIBC_2.1 等的版本！

检验软件正确性

前面提到很多升级与安装需要注意的事项，因为我们需要克服很多的程序漏洞，所以需要前往 Linux distribution 或者是某些套件开发商的网站，下载最新并且较安全的档案来安装才行。 好了，那么『有没有可能我们下载的档案本身就有问题？』 是可能的！因为骇克无所不在，很多的套件开发商已经公布过他们的网页所放置的档案曾经被窜改过！ 那怎么办？连下载原版的数据都可能有问题了？难道没有办法判断档案的正确性吗？！

这个时候我们就要透过每个档案独特的指纹验证数据了！因为每个档案的内容与档案大小都不相同， 所以如果一个档案被修改之后，必然会有部分的信息不一样！利用这个咚咚，我们可以使用 MD5 这个指纹验证机制来判断该档案有没有被更动过！举个例子来说，义守大学提供的 Red Hat 9 原版光盘下载点(http://ftp.isu.edu.tw/pub/Linux/RedHat/linux/9/en/iso/i386/)同时提供了 Red Hat 9 所有光盘 ISO 档案的 MD5 编码，透过这个编码的比对， 我们就可以晓得下载的档案是否有问题。那么万一 Red Hat

提供的光盘映象文件(image)被下载之后，让有心人士偷偷修改过，再转到 Internet 上面流传，那么你下载的这个档案偏偏不是原厂提供的，呵呵！ 你能保证该档案的内容完全没有问题吗？！当然不能对不对？！是的，这个时候就有 md5sum 这个档案指纹的咚咚出现啦！说说他的用法吧！

---

md5sum

我们以 Red Hat 在 2004 年发布的一则程序臭虫修订程序为例：

https://rhn.redhat.com/errata/RHBA-2004-083.html

这个 grep-2.5.1-7.8.i386.rpm 的档案他的 MD5 指纹编码是：『5a0c3fcfcd4c3f937644b8cd71a0cf89』，如果您下载了这个档案，并且执行底下的指令，应该得到相同的指纹码的：

```
[root@linux ~]# md5sum [-bct] filename
[root@linux ~]# md5sum [--status|--warn] --check filename
参数：
-b ：使用 binary 的读档方式，预设为 Windows/DOS 档案型态的读取方式；
-c ：检验 md5sum 档案指纹；
-t ：以文字型态来读取 md5sum 的档案指纹。
范例：

范例一：将刚刚的档案下载后，测试看看！
[root@linux ~]# wget \
> ftp://updates.redhat.com/9/en/os/i386/grep-2.5.1-7.8.i386.rpm
[root@linux ~]# md5sum grep-2.5.1-7.8.i386.rpm
5a0c3fcfcd4c3f937644b8cd71a0cf89  grep-2.5.1-7.8.i386.rpm
# 看！显示的编码是否与上面相同呢？！赶紧测试看看！
```

一般而言，每个系统里面的档案内容大概都不相同，例如你的系统中的 /etc/passwd 这个登入信息文件与我的一定不一样，因为我们的使用者与密码、 Shell 及家目录等大概都不相同，所以由 md5sum 这个档案指纹分析程序所自行计算出来的指纹表当然就不相同啰！

好了，那么如何使用这个东西呢？基本上，您必须要在您的 Linux 系统上为您的这些重要的档案进行指纹数据库的建立(好像在做户口调查！)，将底下这些档案建立数据库：

- /etc/passwd
- /etc/shadow(假如你不让使用者改密码了)
- /etc/group
- /usr/bin/passwd
- /sbin/portmap
- /bin/login (这个也很容易被骇！)
- /bin/ls
- /bin/ps
- /usr/bin/top

等等，这几个档案最容易被修改了！因为很多木马程序执行的时候，还是会有所谓的『执行序, PID』为了怕被 root 追查出来，所以他们都会修改这些检查排程的档案，如果你可以替这些档案建立指纹数据库(就

是使用 md5sum 检查一次，将该档案指纹记录下来，然后常常以 shell script 的方式由程序自行来检查指纹表是否不同了！)，那么对于档案系统会比较安全啦！！

重点回顾

- 原始码其实大多是纯文字文件，需要透过编译器的编译动作后，才能够制作出 Linux 系统能够认识的可执行的 binary file ；
- 在 Linux 系统当中，最标准的 C 语言编译器为 gcc ；
- 在编译的过程当中，可以藉由其它套件提供的函式库来使用该套件的相关机制与功能；
- 为了简化编译过程当中的复杂的指令输入，可以藉由 make 与 makefile 规则定义，来简化程序的更新、编译与连结等动作；
- Tarball 为使用 tar 与 gzip 压缩功能所打包与压缩的，具有原始码程序文件的档案；
- 一般而言，要使用 Tarball 管理 Linux 系统上的套件，最好需要 gcc, make, autoconfig, kernel source, kernel header 等前驱套件才行，所以在安装 Linux 之初，最好就能够选择 Software development 以及 kernel development 之类的群组；
- 函式库有动态函式库与静态函式库，动态函式库在升级上具有较佳的优势。动态函式库的附档名为 *.so 而静态则是 *.a ；
- patch 的主要功能在更新原始码，所以更新原始码之后，还需要进行重新编译的动作才行；
- 可以利用 ldconfig 与 /etc/ld.so.conf 来制作动态函式库的连结与快取！
- 透过 MD5 的编码可以判断下载的档案是否为原本厂商所释出的档案。

参考资源

如果您对于程序的开发相当的有兴趣，那么真的建议挑这个跨平台的 C 语言来学习！

- gcc 的使用简介：http://zope.slat.org/Members/ycheng/Document/gcc
- gdb 的使用简介：http://zope.slat.org/Members/ycheng/Document/gdb
- C 程序语言：http://www.cyut.edu.tw/~ckhung/b/c/

课后练习

RPM 与 SRPM 套件管理员

最近更新日期：2005/10/03

在上一章当中，我们介绍了以 Tarball 的方式来安装我们的套件，同时也说明了 Source code 与执行档之间的关系。不过，如果每次安装套件都需要进行编译的动作，那么实在很没效率！这个时候，由 Red Hat 公司所开发出来的套件管理程序(Red Hat Package Manager, RPM)可就帮了大忙了！RPM 除了可以用来安装套件之外，还可以进行查询、升级、反安装、以及其它验证等等的功能， 这些功能让我们在管理系统的套件上，更显的方便呢！此外，我们也可以利用 RPM 的原理来『自行创造自己的 RPM 档案』呢！

由于 RPM 实在是太好用了，目前主要的 Linux distributions 都是使用 RPM 来管理他们的套件，例如 Red Hat 系统 (含 Fedora), SuSE 与改版后的 Mandriva ，所以，您不能不知道 RPM 是什么东西？该如何利用他，以及熟悉相关的功能！赶紧来参详参详！

前言

在前一章我们提到以原始码的方式来安装套件，也就是利用厂商释出的 Tarball 来进行套件与程序的安装。不过，您应该很容易发现，那就是每次安装套件都需要设定操作系统、 设定编译参数、实际的编译、最后还要依据个人喜好的方式来安装套件到定位。 这过程是真的很麻烦的，而且对于不熟整个系统的朋友来说，还真是累人啊！

那有没有想过，如果我的 Linux 系统与厂商的系统一模一样，那么在厂商的系统上面编译出来的执行档，自然也就可以在我的系统上面跑啰！也就是说，厂商先在他们的系统上面编译好了我们使用者所需要的套件， 然后将这个编译好的可执行的套件直接释出给使用者来安装，如此一来，由于我们本来就使用厂商的 Linux distribution ，所以当然系统是一样的，那么使用厂商提供的编译过的可执行档就没有问题啦！ 说的比较白话一些，那就是利用类似 Windows 的安装方式，由程序开发者直接在已知的系统上面编译好，再将该程序直接给使用者来安装，如此而已。

那么如果在安装的时候还可以加上一些与这些程序相关的信息，将他建立成为数据库， 那不就可以进行安装、反安装、升级与验证等等的相关功能啰(类似 Windows 底下的『新增移除程序』)？！确实如此，在 Linux 上面至少就有两种常见的这方面的套件管理员，分别是 RPM 与 Debian 的 dpkg ，其中又以 RPM 更常见。所以底下我们就来介绍一下 RPM 这个咚咚啰！

什么是 RPM 与 SRPM

RPM 全名是『 RedHat Package Manager 』简称则为 RPM 啦！顾名思义，当初这个套件管理的程序是由 Red Hat 这家公司发展出来的， 但其实在很多的其它套件也有相类似的套件管理程序。不过由于 RPM 使用上很方便，所以就成了目前最热门的套件管理程序啦！

那么什么是 RPM 呢？说的简单一点， RPM 是以一种数据库记录的方式来将你所需要的套件安装到你的 Linux 主机的一套管理程序。他最大的特点就是将您要安装的套件先编译过(如果需要的话)并且打包好了，透过包装好的套件里头预设的数据库记录， 记录这个套件要安装的时候必须要的相依属性模块(就是你的 Linux 主机需要先存在的几个必须的套件)，当安装在你的 Linux 主机时， RPM 会先依照套件里头的纪录数据查询 Linux 主机的相依属性套件是否满足， 若满足则予以安装，若不满足则不予安装。那么安装的时候就将该套件的信息整个写入 RPM 的数据库中，以便未来的查询、验证与反安装！这样一来的优点是：

1. 由于已经编译完成并且打包完毕，所以安装上很方便(不需要再重新编译)；
2. 由于套件的信息都已经记录在 Linux 主机的数据库上，很方便查询、升级与反安装；

但是这也造成很大的困扰，由于 RPM 程序是已经包装好的数据，也就是说， 里面的数据已经都『编译完成』了！所以， 安装的时候一定需要当初安装时的主机环境才能安装 ，也就是说，当初建立这个套件的安装环境必须也要在你的主机上面出现才行！例如 rp-pppoe 这个 ADSL 拨接套件，他必须要在 ppp 这个套件存在的环境下才能进行安装！如果你的主机并没有 ppp 这个套件，那么很抱歉，除非您先安装 ppp 否则 rp-pppoe 就是不让你安装的(当然您可以强制安装，但是通常都会有点问题发生就是了！)。

所以，通常不同的 distribution 所释出的 RPM 档案，并不能用在其它的 distributions 里面，举例来说， Fedora 释出的 RPM 档案，通常无法直接在 Mandriva 上面进行安装的，更有甚者， 不同版本之间也无法互通，例如 Fedora Core 4 的 RPM 档案就无法直接套用在 FC3 上面！因此，这样可以发现他的缺点是：

1. 安装的环境必须与打包时的环境需求一致或相当；
2. 需要满足套件的相依属性需求；
3. 反安装时需要特别小心，最底层的套件不可先移除，否则可能造成整个系统的问题！

那怎么办？呵呵！还好，还有 SRPM 这个东西！ SRPM 是什么呢？顾名思义，他是 Source RPM 的意思，也就是这个 RPM 档案里面含有原始码(Source Code)哩！特别注意的是，这个 SRPM 所提供的套件内容『并没有经过编译』， 他提供的是原始码喔！

通常 SRPM 的附檔名是以 ***.src.rpm 这种格式来命名的 。不过，既然 SRPM 提供的是原始码，那么为什么我们不使用 Tarball 直接来安装就好了？！这是因为 SRPM 虽然内容是原始码， 但是他仍然含有该套件所需要的相依性套件说明、以及所有 RPM 档案所提供的数据，同时，他与 RPM 不同的是，他也提供了参数设定档(就是 configure 与 makefile 啦！)。所以，如果我们下载的是 SRPM ，那么要安装该套件时，RPM 套件管理员将会(1)先将该套件以 RPM 管理的方式编译，(2)然后将编译完成的 RPM 档案安装到 Linux 系统当中。与 RPM 档案相比，SRPM 多了一个重新编译的动作，而且 SRPM 编译完成会产生 RPM 档案。

怪了，怎么 SRPM 这么麻烦呐！还要重新编译一次，那么我们直接使用 RPM 来安装不就好了！？通常一个套件在释出的时候，都会同时释出该套件的 RPM 与 SRPM 。我们现在知道 RPM 档案必须要在相同的 Linux 环境下才能够安装，而 SRPM 既然是原始码的格式，自然我们就可以透过修改 SRPM 内的参数设定档，然后重新编译产生能适合我们 Linux 环境的 RPM 档案，如此一来，不就可以将该套件安装到我们的系统当中，而不必与原作者打包的 Linux 环境相同了？这就是 SRPM 的用处了！

什么是 i386, i586, i686, noarch

好啦！现在我们已经知道 RPM 与 SRPM 的格式了，分别为：

```
xxxxxxxxx.rpm   <==RPM 的格式，已经经过编译且包装完成的 rpm 档案；
xxxxx.src.rpm   <==SRPM 的格式，包含未编译的原始码信息。
```

那么我们怎么知道这个套件的版本、适用的平台、打包的次数呢？呵呵！只要透过档名就可以知道了！例如 rp-pppoe-3.1-5.i386.rpm 这的档案的意义为：

```
rp-pppoe -       3.1     -      5        .i386        .rpm
套件名称    套件的版本信息 释出的次数 适合的硬件平台 附文件名
```

除了后面适合的硬件平台与附文件名外，主要是以『-』来隔开各个部分， 这样子可以很清楚的发现该套件的名称、版本信息、打包次数与操作的硬件平台！ 好了，来谈一谈每个不同的地方吧：

- 套件名称：
  当然就是每一个套件的名称了！上面的范例就是 rp-pppoe 。
- 版本信息：
  每一次更新版本就需要有一个版本的信息，否则如何知道这一版是新是旧？ 这里通常又分为主版本跟次版本，以上面为例，主版本为 3 ，在主版本的架构下更动部分原始码内容，而释出一个新的版本，就是次版本啦！ 以上面为例，就是 1 啰！
- 释出版本次数：
  也就是编译的次数啦！那么为何需要重复的编译呢？ 这是由于同一版的套件中，可能由于有某些 bug 或者是安全上的顾虑，所以必须要重新设定当初打包时候的设定参数， 设定完成之后重新编译并打包成 RPM 档案！因此就有不同的打包数出现了！(注：这个时候原始码其实还是 3.1 那个版本，只是下达编译时的参数不同而已！)

- 操作硬件平台：
  这是个很好玩的地方，由于 RPM 可以适用在不同的操作平台上， 但是由于不同的平台设定的参数还是有所差异性！并且，我们可以针对比较高阶的 CPU 来进行最佳化参数的设定，所以就有所谓的 i386, i586, i686 与 noarch 等的文件名称出现了！

平台名称	适合平台说明
i386	几乎适用于所有的 x86 平台，不论是旧的 pentum 或者是新的 pentum-IV 与 K7 系列的 CPU 等等，都可以正常的工作！那个 i 指的是 Intel 兼容的 CPU 的意思，至于 386 不用说，就是 CPU 的等级啦！
i586	就是 586 等级的计算机，那是哪些呢？包括 pentum 第一代 MMX CPU， AMD 的 K5, K6 系列 CPU (socket 7 插脚) 等等的 CPU 都算是这个等级；
i686	在 pentun II 以后的 Intel 系列 CPU ，及 K7 以后等级的 CPU 都属于这个 686 等级！
noarch	就是没有任何硬件等级上的限制。一般来说，这种类型的 RPM 档案，里面应该没有 binary file 存在。

-
  需要额外说明的是，i386 的档案可以在任何的机器上面安装，不论是 586 或者是 686 的机器，但是 i686 则不一定可以使用于 386 或者是 586 的硬件上面，这是因为 i686 的 RPM 档案在编译的时候，主要是针对 686 硬件等级的 CPU 来进行最佳化编译，而 386/586 等级的硬件可能由于无法支持该最佳化参数， 所以无法使用呢！另外，在 686 的机器上使用 i686 的档案会比使用 i386 的档案，效能可能比较好一些！无论如何，使用 i386 应该就是比较没有问题的啦！另外，由于不同的 distribution 会有不同的环境与函式库，所以在 i386 之后也有可能会额外再加上该套件的简写！

RPM 的优点

RPM 有以下的优点：

- RPM 档案本身为已经编译过的 binary 档案，可以让 client 端的使用者免除重新编译的困扰；
- RPM 档案在被安装之前，RPM 会先检查系统的硬盘容量、操作系统版本等，可避免档案被安装错误；
- RPM 档案本身提供套件版本信息、相依属性套件名称、套件用途说明、套件所含档案等信息，便于了解套件；
- RPM 管理的方式使用数据库记录 RPM 档案的相关参数，便于升级、移除、查询与验证。

为什么 RPM 在使用上很方便呢？我们前面提过， RPM 这个套件管理员所处理的套件，是由套件提供者在特定的 Linux 作业平台上面将该套件编译完成，并且打包好，那使用者只要拿到这个打包好的套件， 然后将里头的档案放置到应该要摆放的目录，不就完成安装啰？！对啦！就是这样！ 但是有没有想过，我们在前一章原始码与 Tarball 里面提过的，有些套件是有相关性的，例如要安装网络卡驱动程序，就得要有 kernel source 与 gcc 及 make 等套件。那么我们的 RPM 套件是否一定可以安装完成呢？！ 如果该套件安装之后，却找不到他相关的前驱套件，那不是挺麻烦的吗？因为安装好的套件也无法使用啊！

为了解决这种具有相关性套件之间的问题，就是所谓的套件相依属性，RPM 就在提供套件打包的档案时，同时加入一些讯息登录的功能，这些讯息包括套件的版本、 打包套件者、相依属性的套件、套件的功能说明、该套件的所有档案与目录纪录、等等，然后在 Linux 系统上面亦建立一个 RPM 套件数据库(database)，如此一来，当您要安装某个以 RPM 型态提供的套件时，在安装的过程中， RPM 会去检验一下数据库里面是否已经存在相关的套件了， 如果数据库显示不存在，那么这个 RPM 档案『预设』就不能安装(会显示一些错误讯息)。呵呵！没有错，这个就是 RPM 类型的档案最为人所诟病的『 套件的属性相依』问题啦！

RPM 属性相依的克服方式

虽然 RPM 有套件属性相依的问题，但是 RPM 的优点实在是比缺点要好的多，所以很多使用者还是偏好使用 RPM 来管理自己的套件，在这样的情况下，如何解决 RPM 的属性相依问题呢？最简单的方式就是在安装 RPM 档案的时候，发生套件相依的问题时，手动去下载前驱套件，先安装好这些套件， 然后再安装最终想要安装的套件即可。但是，如此一来很花费使用者的精神与时间，挺麻烦的啦！ 有没有比较快速的方法呢？

呵呵！有的，由于 RPM 类型的档案里面含有属性相依的讯息存在，如果我们可以透过分析这些讯息， 再让程序自行去寻找未安装的前驱套件，并事先加以安装，如此一来不就解决了属性相依的困扰了吗？！ 没错！是这样！这就是目前所谓的 urpmi/apt/yum 等服务的由来啦！这些服务都是透过分析 RPM 档案的相依信息，然后自行取得相依属性套件， 自行完成安装的动作，呵呵！相当的方便呢！这些信息我们会在 服务器架设篇 里面进行介绍，在这个章节当中，我们主要还是以单纯的 RPM 为主喔！

RPM 套件管理程序

RPM 的使用其实不难，只要使用 rpm 这个指令即可！鸟哥最喜欢的就是 rpm 指令的查询功能了，可以让我很轻易的就知道某个系统有没有安装鸟哥要的套件呢！此外， 我们最好还是得要知道一下，到底 RPM 类型的档案他们是将套件的相关档案放置在哪里呢？还有，我们说的那个 RPM 的数据库又是放置在哪里呢？

RPM 预设安装的路径

一般来说，RPM 类型的档案在安装的时候，会先去读取档案内记载的设定参数内容，然后将该数据用来比对 Linux 系统的环境(例如属性相依的套件)，例如目前 SSH 这个远程联机软件(请参考服务器篇)使用的是 OpenSSL 的加密机制，所以，要安装 SSH 的时候，就得要先安装好 OpenSSL 才行啊，如果没有安装 OpenSSL 的话， SSH 就不让您安装了！这些都是 RPM 环境的要求， 如果环境相符就予以安装，如果不符就会显示出不符合的内容所在！ 等到安装完毕之后， rpm 就会将套件的信息写入：/var/lib/rpm 这个目录中去！所以， 往后您在进行查询的时候或者是预计要升级的时候，相关的信息就会由 /var/lib/rpm 这个目录的内容数据来提供啰！

一般来说，由于 RPM 有数据库来纪录套件相关的信息，所以 RPM 类型的套件所拥有的档案都放置在系统预设放置的目录底下，亦即如同我们在 档案属性与目录配置 一文当中提到的：

/etc	一些设定文件放置的目录，例如 /etc/crontab
/usr/bin	一些可执行档案
/usr/lib	一些程序使用的动态函式库
/usr/share/doc	一些基本的软件使用手册与说明文件
/usr/share/man	一些 man page 档案

好了，底下我们就来针对每个 RPM 的相关指令来进行说明啰！

RPM 安装(install)

安装就是 install 嘛！所以啰，使用 rpm 来安装就很简单啦！假设我要安装一个档名为 rp-pppoe-3.1-5.i386.rpm 的档案，那么我可以这样(记得某些套件可能需要以系统管理员的身份来安装)：

```
[root@linux ~]# rpm -i rp-pppoe-3.1-5.i386.rpm
```

不过，这样的参数其实无法显示安装的进度，所以，通常我们会这样下达安装指令：

```
[root@linux ~]# rpm -ivh package_name
参数：
-i ：install 的意思
-v ：察看更细部的安装信息画面
-h ：以安装信息列显示安装进度
范例：

范例一：安装 rp-pppoe-3.1-5.i386.rpm
[root@linux ~]# rpm -ivh rp-pppoe-3.1-5.i386.rpm
Preparing...     ########################################### [100%]
   1:rp-pppoe    ########################################### [100%]

范例二、一口气安装两个以上的套件时：
[root@linux ~]# rpm -ivh a.i386.rpm b.i386.rpm *.rpm
# 后面直接接上许多的套件档案！

范例三、直接由网络上面的某个档案安装，以网址来安装：
[root@linux ~]# rpm -ivh http://website.name/path/pkgname.rpm
```

另外，如果我们在安装的过程当中发现问题，或者已经知道会发生的问题， 而还是『执意』要安装这个套件时，可以使用如下的参数『强制』安装上去：

可下达的参数	代表意义
--nodeps	使用时机： 如果您在安装某个套件时，老是发现 rpm 告诉你『有属性相依的套件尚未安装』， 而您又想要直接强制安装这个套件时，可以加上

	--nodeps 告知 RPM 不要去检查套件的相依性。 危险性： 套件会有相依性的原因是因为彼此会使用到对方的机制或功能，如果强制安装而不考虑套件的属性相依， 则可能会造成该套件的无法正常使用！
--nomd5	使用时机： 不想检查 RPM 档案所含的 MD5 信息时。 说明： 还记得我们在前一章有提到的 MD5 这个指纹辨识吧？！没错，这里指的就是不要检查 RPM 套件的 MD5 信息。但除非您很清楚这个套件的来源，否则不建议使用这个参数。
--noscripts	使用时机： 不想让该套件自行启用或者自行执行某些系统指令。 说明： RPM 的优点除了可以将档案放置到定位之外，还可以自动执行一些前置作业的指令，例如数据库的初始化。 如果您不想要让 RPM 帮您自动执行这一类型的指令，就加上他吧！
--replacefiles	使用时机： 如果在安装的过程当中出现了『某个档案已经被安装在您的系统上面』的信息， 又或许出现版本不合的讯息(confilcting files)时，可以使用这个参数来直接覆盖档案。 危险性： 覆盖的动作是无法复原的！所以，您必须要很清楚的知道被覆盖的档案是真的不重要喔！否则会欲哭无泪！
--replacepkgs	使用时机： 重新安装某个已经安装过的套件！
--force	使用时机： 这个参数其实就是 --replacefiles 与 --replacepkgs 的综合体！
--test	使用时机： 想要测试一下该套件是否可以被安装到使用者的 Linux 环境当中。范例为： rpm -ivh pkgname.i386.rpm --test

一般来说，安装的指令大约就是这些了。通常鸟哥建议直接使用 -ivh 就好了， 如果安装的过程中发现问题，一个一个去将问题找出来，尽量不要使用『 暴力安装法 』，因为可能会发生很多不可预期的问题呢！除非您很清楚的知道使用上面的参数后，安装的结果是您预期的！

RPM 升级与更新

使用 RPM 来升级真是太简单了！就以 -Uvh 或 -Fvh 来升级即可(注：vh 的功能仍是在于显示细部信息与安装进度而已)！不过，这两种升级方式是不太一样的：

-Uvh	后面接的套件即使没有安装过，则系统将予以直接安装； 若后面接的套件有安装过旧版，则系统自动更新至新版；
-Fvh	如果后面接的套件并未安装到您的 Linux 系统上，则该套件不会被安装；亦即只有安装至您 Linux 系统内的套件会被『升级』！

由上面的说明来看，如果您想要大量的升级系统旧版本的套件时(例如刚安装完操作系统，而想要更新套件至最新)，使用 -Fvh 则是比较好的作法。但是需要注意的是，如果您使用的是 Fvh ，偏偏您的机器上

尚无这一个套件，那么很抱歉，该套件并不会被安装在您的 Linux 主机上面，所以请重新以 ivh 来安装吧！

通常有的朋友在进行整个操作系统的旧版套件修补时，喜欢这么进行：

1. 先到各发展商的 errata 网站或者是国内的 FTP 映像站捉下来最新的 RPM 档案；
2. 使用 -Fvh 来将您的系统内曾安装过的套件进行修补与升级！(真是方便呀！)

当然啰，升级也是可以利用 --nodeps/--force 等等的参数啦！

RPM 查询

RPM 在查询的时候，其实查询的地方是在 /var/lib/rpm 这个目录下的数据库档案啦！另外， RPM 也可以查询档案内的信息喔！那如何去查询呢？我们底下以 简单的范例来说明：

```
[root@linux ~]# rpm -qa
[root@linux ~]# rpm -q[licdR] 已安装的套件名称
[root@linux ~]# rpm -qf 存在于系统上面的某个档案
[root@linux ~]# rpm -qp[licdR] 未安装的某个文件名称
参数：
在查询的部分，所有的参数之前都需要加上 -q 才是所谓的查询！
查询主要分为两部分，一个是查已安装，另一个则是查某个 rpm 档案内容。
查询已安装套件的信息：
-q  ：仅查询，后面接的套件名称是否有安装；
-qa ：列出所有的，已经安装在本机 Linux 系统上面的所有套件名称；
-qi ：列出该套件的详细信息 (information)，包含开发商、版本与说明等；
-ql ：列出该套件所有的档案与目录所在完整文件名 (list)；
-qc ：列出该套件的所有设定档 (找出在 /etc/ 底下的檔名而已)
-qd ：列出该套件的所有说明档 (找出与 man 有关的档案而已)
-qR ：列出与该套件有关的相依套件所含的档案 (Required 的意思)
-qf ：由后面接的文件名称，找出该档案属于哪一个已安装的套件；
查询某个 RPM 档案内含有的信息：
-qp[icdlR]：注意 -qp 后面接的所有参数以上面的说明一致。但用途仅在于找出
              某个 RPM 档案内的信息，而非已安装的套件信息！注意！
范例：

范例一：找出你的 Linux 是否有安装 logrotate 这个套件？
[root@linux ~]# rpm -q logrotate
logrotate-3.7.1-10
[root@linux ~]# rpm -q logrotating
package logrotating is not installed
# 注意到，系统会去找是否有安装后面接的套件名称。注意，
# 不必要加上版本喔！至于显示的结果，一看就知道有没有安装啦！
```

```
范例二：列出上题当中，该套件的所有目录与档案：
[root@linux ~]# rpm -ql logrotate
/etc/cron.daily/logrotate
/etc/logrotate.conf
......以下省略......
# 可以看出该套件到底提供了多少的档案与目录。

范例三：列出 logrotate 这个套件的相关说明资料：
[root@linux ~]# rpm -qi logrotate
Name        : logrotate                Relocations: (not relocatable)
Version     : 3.7.1                         Vendor: Red Hat, Inc.
Release     : 10                        Build Date: Fri Apr  1 03:54:42 2005
Install Date: Sat Jun 25 08:28:26 2005  Build Host: tweety.build.redhat.com
Group       : 系统环境/基础              Source RPM: logrotate-3.7.1-10.src.rpm
Size        : 47825                        License: GPL
Signature   : DSA/SHA1, Sat May 21 01:34:11 2005, Key ID b44269d04f2a6fd2
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary     : 循环、压缩、移除以及邮寄系统纪录档案。
Description :
The logrotate utility is designed to simplify the administration of
log files on a system which generates a lot of log files. Logrotate
allows for the automatic rotation, compression, removal, and mailing of
log files. Logrotate can be set to handle a log file daily, weekly,
monthly, or when the log file gets to a certain size. Normally,
logrotate runs as a daily cron job.
# 列出该套件的 information (信息)，里面的信息可多着呢，包括了套件名称、
# 版本、开发商、SRPM 文件名称、打包次数、简单说明信息、套件打包者、
# 安装日期等等！如果想要详细的知道该套件的数据，用这个参数来了解一下

范例四：分别仅找出 logrotate 的设定档与说明档
[root@linux ~]# rpm -qc logrotate
[root@linux ~]# rpm -qd logrotate

范例五：若要成功安装 logrotate ，他还需要什么档案的帮忙？
[root@linux ~]# rpm -qR logrotate
/bin/sh
config(logrotate) = 3.7.1-10
libc.so.6
....以下省略....
# 由这里看起来，呵呵～还需要很多档案的支持才行喔！

范例六：由上面的范例五，找出 /bin/sh 是那个套件提供的？
[root@linux ~]# rpm -qf /bin/sh
```

```
bash-3.0-31
# 这个参数后面接的可是『档案』呐！不像前面都是接套件喔！
# 这个功能在查询系统的某个档案属于哪一个套件所有的。

范例七：假设我有下载一个 RPM 档案，想要知道该档案的需求档案，该如何？
[root@linux ~]# rpm -qpR filename.i386.rpm
# 加上 -qpR ，找出该档案需求的数据！
```

常见的查询就是这些了！要特别说明的是，在查询本机上面的 RPM 套件相关信息时， 不需要加上版本的名称，只要加上套件名称即可！因为他会由 /var/lib/rpm 这个数据库里面去查询， 所以我们可以不需要加上版本名称。但是查询某个 RPM 档案就不同了， 我们必须要列出整个档案的完整档名才行～ 这一点朋友们常常会搞错。底下我们就来做几个简单的练习吧！

例题：
我想要知道我的系统当中，以 c 开头的套件有几个，如何实做？

rpm -qa | grep ^c | wc -l

我的 WWW 服务器为 Apache ，我知道他使用的 RPM 套件档名为 httpd 。现在，我想要知道这个套件的所有设定档放置在何处，可以怎么作？

rpm -qc httpd

承上题，如果查出来的设定档案已经被我改过，但是我忘记了曾经修改过哪些地方，所以想要直接重新安装一次该套件，该如何作？

假设该套件在网络上的网址为：
http://web.site.name/path/httpd-x.x.xx.i386.rpm
则我可以这样做：
rpm -ivh http://web.site.name/path/httpd-x.x.xx.i386.rpm --replacepkgs

如果我误砍了某个重要档案，例如 /etc/crontab，偏偏不晓得他属于哪一个套件，该怎么办？！

虽然已经没有这个档案了，不过没有关系，因为 RPM 有纪录在 /var/lib/rpm 当中的数据库啊！所以直接下达：
rpm -qf /etc/crontab
就可以知道是那个套件啰！重新安装一次该套件即可！

RPM 验证与数位签章

验证的功能主要在于提供系统管理员一个有用的管理机制！作用的方式是『 使用 /var/lib/rpm 底下的数据库内容来比对目前 Linux 系统的环境下的所有套件档案 』也就是说，当您有数据不小心遗失， 或者是因为您误杀了某个套件的档案，或者是不小心不知道修改到某一个套件的档案内容， 就用这个简单的方法来验证一下原本的档案系统吧！ 好让您了解这一阵子到底是修改到哪些档案数据了！验证的方式很简单：

```
[root@linux ~]# rpm -Va
[root@linux ~]# rpm -V  已安装的套件名称
[root@linux ~]# rpm -Vp 某个 RPM 档案的档名
[root@linux ~]# rpm -Vf 在系统上面的某个档案
参数：
```

```
-V  ：后面加的是套件名称，若该套件所含的档案被更动过，才会列出来；
-Va ：列出目前系统上面所有可能被更动过的档案；
-Vp ：后面加的是文件名称，列出该套件内可能被更动过的档案；
-Vf ：列出某个档案是否被更动过～
范例：

范例一：列出你的 Linux 内的 logrotate 这个套件是否被更动过？
[root@linux ~]# rpm -V logrotate
# 如果没有出现任何讯息，恭喜你，该套件没有被更动过。
# 如果有出现任何讯息，才是有出现状况啊！

范例二：查询一下，你的 /etc/crontab 是否有被更动过？
[root@linux ~]# rpm -Vf /etc/crontab
S.5....T  c /etc/crontab
# 瞧！因为有被更动过，所以会列出被更动过的信息！
```

好了，那么我怎么知道到底我的档案被更动过的内容是什么？呵呵！简单的说明一下吧！ 例如，我们检查一下 logrotate 这个套件：

```
[root@linux ~]# rpm -ql logrotate
/etc/cron.daily/logrotate
/etc/logrotate.conf
/etc/logrotate.d
/usr/sbin/logrotate
/usr/share/doc/logrotate-3.7.1
/usr/share/doc/logrotate-3.7.1/CHANGES
/usr/share/man/man8/logrotate.8.gz
/var/lib/logrotate.status
# 呵呵！共有八个档案啊！

[root@linux ~]# rpm -V logrotate
..5....T c /etc/logrotate.conf
# 上面的信息是这样的：
S ：file Size differs
    档案的容量大小是否被改变
M ：Mode differs (includes permissions and file type)
    档案的类型或档案的属性，如是否可执行等参数已被改变
5 ：MD5 sum differs
    MD5 这一种加密防骇的属性已被改变
D ：Device major/minor number mis-match
    装置名称已被改变
L ：readLink(2) path mis-match
    Link 属性已被改变
U ：User ownership differs
    档案的所属人已被改变
```

```
G : Group ownership differs
    档案的所属群组已被改变
T : mTime differs
    档案的建立时间已被改变
```

所以，如果当一个档案所有的信息都被更动过，那么他的显示就会是：

```
SM5DLUGT c filename
```

至于那个 c 代表的是『 Config file 』的意思，也就是档案的类型，档案类型有底下这几类：

- c ：设定档(config file)
- d ：文件数据文件(documentation)
- g ：鬼档案～通常是该档案不被某个套件所包含，较少发生！(ghost file)
- l ：授权档案(license file)
- r ：自述文件(read me)

经过验证的功能，您就可以知道那个档案被更动过。那么如果该档案的变更是『预期中的』， 那么就没有什么大问题，但是如果该档案是『非预期的』，那么是否被入侵了呢？呵呵！ 得注意注意啰！

再来，由于数字签证的盛行，我们 Linux 的 RPM 也可以利用数字签证来判断待安装的套件档案是否有问题喔！一般我们使用的是 GPG 的金钥(public key)。应用的方法很简单，首先， 当我们想要使用某个团体释出的套件时，就需要将他们释出的 GPG 金钥先安装在自己的 Linux 系统上。然后，当安装该团体释出的套件时，就会检查两者的 key 是否相同，如果相同就直接安装，如果不同就会在屏幕上面显示讯息告知您并未安装该团体的 GPG 金钥！

安装金钥的方法很简单，例如 Red Hat 本身就有金钥在系统当中，安装如下：

```
[root@linux ~]# rpm --import /usr/share/rhn/RPM-GPG-KEY
```

一般来说，您的 Linux distributions 都会释出自己的 GPG Key 的，如果是 Red Hat 系统的话， 可以使用：

```
[root@linux ~]# locate GPG-KEY
```

来进行搜寻档案的工作，至于这些金钥的内容，我们可以这样查询喔：

```
[root@linux ~]# rpm -qa | grep gpg
libgpg-error-1.0-2
gpg-pubkey-4f2a6fd2-3f9d9d3b
[root@linux ~]# rpm -qi gpg-pubkey-4f2a6fd2-3f9d9d3b
Name        : gpg-pubkey                  Relocations: (not relocatable)
Version     : 4f2a6fd2                         Vendor: (none)
Release     : 3f9d9d3b                     Build Date: Sat Jun 25 22:13:00 2005
Install Date: Sat Jun 25 22:13:00 2005     Build Host: localhost
Group       : Public Keys                  Source RPM: (none)
Size        : 0                               License: pubkey
Signature   : (none)
```

```
Summary     : gpg(Fedora Project <fedora@redhat.com>)
Description :
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: rpm-4.4.1 (beecrypt-3.0.0)

mQGiBD+dnTsRBACwnlz4AhctOLlVBAsq+RaU82nb5P3bD1YJJpsAce1Ckd2sBUOJD11NUCqH
.....中间省略.....
=mJAx
-----END PGP PUBLIC KEY BLOCK-----
```

这样就能看到相关的信息啰！ ^_^

RPM 反安装与重建数据库

反安装就是将套件解除安装啦！要注意的是，『解安装的过程一定要由最上层往下解除』，以 rp-pppoe 为例，这一个套件主要是依据 ppp 这个套件来安装的，所以当您要解除 ppp 的时候，就必须要先解除 rp-pppoe 才行！否则就会发生结构上的问题啦！这个可以由建筑物来说明， 如果你要拆除五、六楼，那么当然要由六楼拆起，否则拆了第五楼，那么上面的楼层难道会悬空？

那么重建数据库呢？由于我们会一直在修改一些档案内容，例如 /etc/xinetd.d 里头的参数档案，加上可能自系统操作的过程中新增、移除等等的动作，导致系统的数据库有点乱， 这个时候可以使用--rebuilddb 来重建一下 rpm 的数据库！这两个方法的参数如下啰：

```
[root@linux ~]# rpm -e logrotate   <==解安装 logrotate 套件
[root@linux ~]# rpm --rebuilddb    <==重建数据库
```

SRPM 的使用

谈完了 RPM 类型的套件之后，再来我们谈一谈包含了 Source code 的 SRPM 该如何使用呢？！假如今天我们由网络上面下载了一个 SRPM 的档案，该如何安装他？又，如果我想要修改这个 SRPM 里面原始码的相关设定值，又该如何订正与重新编译呢？！此外，最需要注意的是，新版的 rpm 已经将 RPM 与 SRPM 的指令分开了，SRPM 使用的是 rpmbuild 这个指令，而不是 rpm 喔！如果您是 Red Hat 7.3 以前的用户，那么请使用 rpm 来替代 rpmbuild 啦！

利用系统默认值安装 SRPM 档案

假设我下载了一个 SRPM 的档案，又不想要修订这个档案内的原始码与相关的设定值， 那么我可以直接编译并安装吗？当然可以！利用 rpmbuild 配合参数即可。参数主要有底下两个：

--rebuild	这个参数会将后面的 SRPM 进行『编译』与『打包』的动作，最后会产生 RPM 的档案，但是产生的 RPM 档案并没有安装到系统上。当您使用 --rebuild 的时候，最后通常会发现一行字体： Wrote: /usr/src/RPM/RPMS/i386/pkgname.i386.rpm

	这个就是编译完成的 RPM 档案啰！那么这个档案就可以用来安装啦！安装的时候请加绝对路径来安装即可！
--recompile	这个动作会直接的『编译』『打包』并且『安装』啰！请注意， rebuild 仅『编译并打包』而已，而 recompile 不但进行编译跟打包，还同时进行『安装』了！

一般来说，如果编译的动作顺利的话，那么编译过程所产生的中间暂存盘都会被自动删除， 如果发生任何错误，则该中间档案会被保留在系统上，等待使用者的除错动作！那么， 该如何除错呢？！如果想要自行除错，就得要知道利用 SRPM 的时候，系统会动用到哪些重要的目录了！底下我们就来谈一谈当处理 SRPM 时，系统会使用到的目录。

SRPM 使用的路径与需要的套件

SRPM 既然含有 source code ，那么其中必定有设定档啰，所以首先我们必需要知道，这个 SRPM 在进行编译的时候，会使用到哪些目录呢？这样一来才能够来修改嘛！你可以到你的 /usr/src 这个目录里面去查看一下，通常每个 distribution 提供的目录都不太相同，以 FC4 为例，他是以 /usr/src/redhat/ 为工作目录， Openlinux 则是以 /usr/src/openlinux 为工作目录！无论如何，反正就是在 /usr/src 这个目录下就对了！好了到 /usr/src/redhat 里头去看一看呦：

/usr/src/redhat/SPEC	这个目录当中放置的是该套件的设定档，例如这个套件的信息参数、设定项目等等都放置在这里；
/usr/src/redhat/SOURCE	这个目录当中放置的是该套件的原始档（*.tar.gz 的档案）以及 config 这个设定档；
/usr/src/redhat/BUILD	在编译的过程中，有些暂存的数据都会放置在这个目录当中；
/usr/src/redhat/RPMS	经过编译之后，并且顺利的编译成功之后，将打包完成的档案放置在这个目录当中。里头有包含了 i386, i586, i686, noarch.... 等等的次目录。

此外，在编译的过程当中，可能会发生不明的错误，或者是设定的错误，这个时候就会在 /tmp 底下产生一个相对应的错误档，您可以根据该错误档进行除错的工作呢！ 等到所有的问题都解决之后，也编译成功了，那么刚刚解压缩之后的档案，就是在 /usr/src/redhat/SPEC, SOURCE, BUILD 等等的档案都会被杀掉，而只剩下放置在 /usr/src/redhat/RPMS 底下的档案了！

由于 SRPM 需要重新编译，而编译的过程当中，我们至少需要有 make 与其相关的程序，及 gcc, c, c++ 等其它的编译用的程序语言来进行编译，所以， 如果您在安装的过程当中没有选取软件开发工具之类的套件，呵呵！得重新拿出你的光盘， 然后再安装喔！哈哈！只是得要克服一大堆的属性相依的问题就是了～ 嗯！还是建议您再次的看一下如何安装吧！

设定档的主要内容

刚刚我们在上面提过了，SRPM 还可以更改一些设定的内容，那么要如何修改这些设定的内容呢？我们以简单的 rp-pppoe 这个套件来说明好了。比较可惜的是， rp-pppoe 的官方网站目前 (2005/10) 似乎不再提

供新的 SRPM 档案了，所以，我们是由 rpmfind.net (http://rpmfind.net/) 找到给 FC4 使用的 SRPM 档案 (是测试版喔！)， 你可以自行查阅一下这个 rp-pppoe 相关的信息以及 rpmfind.net 网站提供的信息。

- rp-pppoe 的官方网站：
  http://www.roaringpenguin.com/pppoe/
- rpmfind.net 与 FC 系列有关的 rp-pppoe 说明与下载点：
  http://rpmfind.net//linux/RPM/fedora/devel/src/rp-pppoe-3.5-30.src.html
  ftp://rpmfind.net/linux/fedora/core/development/SRPMS/rp-pppoe-3.5-30.src.rpm

至于基本的过程如下：(鸟哥在这里假设你已经将 rp-pppoe-3.5-30.src.rpm 下载到 /root 底下了)

```
[root@linux ~]# rpm -i /root/rp-pppoe-3.5-30.src.rpm
# 这个过程不会显示任何东西，他只会将 SRPM 的档案解开后，放置到
# /usr/src/redhat 下！

[root@linux ~]# find /usr/src/redhat/ -type f
/usr/src/redhat/SOURCES/rp-pppoe-3.5-buildroot.patch
/usr/src/redhat/SOURCES/adsl-stop
/usr/src/redhat/SOURCES/adsl-start
/usr/src/redhat/SOURCES/adsl-setup
/usr/src/redhat/SOURCES/rp-pppoe-3.4-redhat.patch
/usr/src/redhat/SOURCES/adsl-status
/usr/src/redhat/SOURCES/rp-pppoe-3.5-firewall.patch
/usr/src/redhat/SOURCES/adsl-connect
/usr/src/redhat/SOURCES/rp-pppoe-3.5.tar.gz
/usr/src/redhat/SPECS/rp-pppoe.spec
# 主要含有原始码与一个重要的设定档啊！ rp-pppoe.spec ！
```

好了，来看看我们的设定参数档，亦即是在 /usr/src/redhat/SPECS 内的 *.spec 档案啰！

```
[root@linux ~]# cd /usr/src/redhat/SPECS
[root@linux SPECS]# vi rp-pppoe.spec
# 1. 首先，这个部分在介绍整个套件的基本相关信息！不论是版本还是释出次数等。
Summary: A PPP over Ethernet client (for xDSL support).
Name: rp-pppoe
Version: 3.5
Release: 30
License: GPL
Group: System Environment/Daemons
Url: http://www.roaringpenguin.com/pppoe/
Source: http://www.roaringpenguin.com/rp-pppoe-%{version}.tar.gz
Source1: adsl-connect
Source2: adsl-setup
.....中间省略.....
```

```
# 2. 这部分则是在设定相依属性需求的地方！
Prereq: /sbin/chkconfig
Prereq: /sbin/service
Prereq: fileutils
Requires: ppp >= 2.4.2
Requires: initscripts >= 5.92
Requires: iproute >= 2.6
ExcludeArch: s390 s390x

%description
PPPoE (Point-to-Point Protocol over Ethernet) is a protocol used by
many ADSL Internet Service Providers. This package contains the
Roaring Penguin PPPoE client, a user-mode program that does not
require any kernel modifications. It is fully compliant with RFC 2516,
the official PPPoE specification.

# 3. 编译前的预处理，以及编译过程当中所需要进行的指令，都写在这里
#    尤其 %build 底下的数据，几乎就是 makefile 里面的信息啊！
%prep
%setup -q
%patch0 -p1 -b .config
%patch1 -p1 -b .buildroot
%patch2 -p1 -b .ipchains

%build
cd src
autoconf
CFLAGS="-D_GNU_SOURCE" %configure
make

install -m 0755 %{SOURCE1} scripts
install -m 0755 %{SOURCE2} scripts
install -m 0755 %{SOURCE3} scripts
install -m 0755 %{SOURCE4} scripts
install -m 0755 %{SOURCE5} scripts

%install
rm -rf $RPM_BUILD_ROOT
.....中间省略.....

# 4. 这里列出，这个套件释出的档案有哪些的意思！
%files
%defattr(-,root,root)
```

```
%doc doc/LICENSE scripts/adsl-connect scripts/adsl-setup scripts/adsl-init
%doc scripts/adsl-start scripts/adsl-status scripts/adsl-stop
%doc configs
%config(noreplace) %{_sysconfdir}/ppp/pppoe-server-options
%config(noreplace) %{_sysconfdir}/ppp/firewall*
/sbin/*
%{_sbindir}/*
%{_mandir}/man?/*

# 5. 列出这个套件的更改历史纪录文件!
%changelog
* Mon Aug 15 2005 Than Ngo <than@redhat.com> 3.5-30
- defaultroute should not overridden #152014
.....中间省略.....
* Wed May 31 2000 Than Ngo <than@redhat.de>
- adopted for Winston.
```

注意到的是 rp-pppoe.sepc 这个档案，这是主要的将 SRPM 编译成 RPM 的设定文件，他的基本规则可以这样看：

1. 整个档案的开头以 Summary 为开始，这部份的设定都是最基础的说明内容；
2. 然后每个不同的段落之间，都以%来做为开头，例如%prep 与%install 等；

我们来谈一谈几个常见的 SRPM 设定段落：

- 系统整体信息方面：

Summary	主要的套件说明，例如上表中，我们说明了他是 ppp 的拨接用途啦！
Name	这个就是套件的名称；
Version	这个是套件的版本信息；
Release	这个是该版本打包的次数说明；
License	这个套件的授权模式，我们是使用 GPL 啦！
Group	这个套件的发展团体名称；
Source	这个套件的来源，如果是网络上下载的套件，通常一定会有这个信息来告诉大家这个原始档的来源！
Url	这个原始码的主要官方网站；
Packager	这个套件是经由谁来打包的呢？
Vender	发展的厂商哪；
ExclusiveArch	这个是说明这个套件的适合安装的硬件，通常预设为 i386，当然，你也可以调整为 i586 啦等等的！
Requires	如果你这个套件还需要其它的套件的支持，那么这里就必需写上来，则当你制作

	成 RPM 之后，系统就会自动的去检查啦！这就是『相依属性』的主要来源啰！

上面几个资料通常都必需要写啦！但是如果你的软件没有相依属性的关系时， 那么就可以不需要那个 Requires 啰！

- %description

将您的套件做一个简短的说明！这个也是必需要的。

- %prep

这部份的设定在于『尚未进行设定或安装之前，你要编译完成的 RPM 帮你事先做的事情』，就是 prepare 的简写啰！那么他的工作事项主要有：

1. 寻找套件所需要的目录是否已经存在？确认用的！
2. 事先建立您的套件所需要的目录，或者事先需要进行的任务；
3. 如果待安装的 Linux 系统内已经有安装的时候可能会被覆盖掉的档案时， 那么就必需要进行备份(backup)的工作了！

大致的工作就是这些啦！

- %setup

这个段落就是在建立我们在 Tarball 当中说明的那个 Makefile 档案啦!所以呢,当然就是执行 ./config 之类的设定档案啰！ 那么如果你要自己新增自己的参数，就可以在这个地方加入你的设定值！ 如果你的软件本身没有这方面的需要，里面就不需要编写内容啰！

- %build

build 就是建立啊！所以当然啰，这个段落就是在谈怎么 make 编译成为可执行的程序啰！

- %install

编译完成 (build) 之后，就是要安装啦！安装就是写在这里，也就是类似 Tarball 里面的 make install 的意思啰！

- %files

这个套件安装的档案都需要写到这里来，当然包括了『目录』喔！ 所以连同目录请一起写到这个段落当中！以备查验呢！^_^

- %changelog

这个主要则是在记录这个套件曾经的更新纪录啰！

好了，那么如果您有自订的信息想要加入的话，就选择你要加入的那个段落，将他修改一下吧！ 例如，如果你在设定 Makefile 的时候，希望能够多一些额外的参数设定，那么就找到 %setup 或 %build 那个段落，将他修改成您所需要的样子，就可以啰！

SRPM 的编译指令

再来呢？嗯！没错，修改完成了，自然就是要将他编译成可以安装的 RPM 档案啦！ 这个时候我们就可以直接在 /usr/src/redhat/SPECS 底下下达：

```
[root@linux ~]# rpmbuild -bb rp-pppoe.spec  <==编译成 RPM 档案
[root@linux ~]# rpmbuild -ba rp-pppoe.spec  <==打包成 SRPM 档案
```

这个时候系统就会这样做：

1. 先进入到 BUILD 这个目录中，在 Fedora 底下就是 /usr/src/redhat/BUILD 这个目录；

2. 依照 *.spec 档案内的 Name 与 Version 设定定义出工作的目录名称， 以我们上面的例子为例，那么系统就会在 BUILD 目录中先删除 rp-pppoe-3.5 的目录，再重新建立一个 rp-pppoe-3.5 的目录，并进入该目录；
3. 在新建的目录里面，针对 SOURCES 目录下的来源档案，也就是 *.spec 里面的 Source 设定的那个档案，以 tar 进行解压缩，以我们这个例子来说，则会在 /usr/src/redhat/BUILD/rp-pppoe-3.5 当中，将 /usr/src/redhat/SOURCES/rp-pppoe-3.5.tar.gz 进行解压缩啦！
4. 然后就开始 %setup 的工作；
5. 再来开始 %build 及 %install 的设定与编译！
6. 最后将完成打包的档案给他放置到该放置的地方去，如果你的规定的硬件是在 i386 的系统，那么最后编译成功的*.i386.rpm 档案就会被放置在/usr/src/RPM/RPMS/i386 里面啰！如果是 i586 那么自然就是 /usr/src/redhat/RPMS/i586 目录下啰！

整个步骤大概就是这样子！最后的结果数据会放置在 RPMS 那个目录底下就对啦！

一个打包自己套件的范例

这个就有趣了！我们自己来编辑一下自己制作的 RPM 怎么样？会很难吗？完全不会！ 这里简单的以一个小例子来说明喔！请注意，这个真的只是一个小例子，所以不要觉得奇怪喔！ 其中，比较需要注意的，由于在上面的步骤说明中，我们知道在将 SRPM 编译成为 RPM 的时候，会以 tar 这支程序来将档案解开，因此，我们在进行来源档案的建立时， 就必需要将他打包成为一个 tar.gz 的 tarball 的档案才行！

假设我们编辑了一支 script，内容是这样：

```
[root@linux ~]# cd /usr/src/redhat/SOURCES
[root@linux SOURCES]# vi showvbird.sh
#!/bin/bash
# This file is just used to demo the RPM packaging.
# the only thing is showing the hostname.
HOST=`/bin/hostname`
/bin/echo $HOST
# 先随便建立一个 shell script ，这个是自己的套件的意思啦！

[root@linux SOURCES]# chmod 755 showvbird.sh
[root@linux SOURCES]# tar -zcvf showvbird.tar.gz showvbird.sh
# 注意喔！务必打包才行啊！
```

上面的动作中，我们编辑了一个 shell script 档案，档名为 showvbird.sh，并且将他打包成为具有 gzip 压缩的 tarball 档案，也就是 showvbird.tar.gz 这样的档案才行！请注意，这个 showvbird.tar.gz 档案『必需』放置在 SOURCES 目录之下！

再来则是要编辑那个很重要的 *.spec 档案啰！你可以这样简单的编写一下：

```
[root@linux SOURCE]# cd /usr/src/redhat/SPECS
[root@linux SPECS]# vi showvbird.spec
Summary:   This is a demo RPM package.
```

```
Name:      showvbird
Version:   1.0
Release:   1
License:   GPL
Group:     VBird's Home
Source:    showvbird.tar.gz   <==记得喔！这里写的是刚刚建立的 tarball
Url:       http://linux.vbird.org
Packager:  VBird

%description
This package is just a demo RPM.

%prep
%setup -c
%install
install -m 755 showvbird.sh /usr/local/bin/showvbird.sh

%files
/usr/local/bin/showvbird.sh
```

好了！开始给他编译并打包成为 RPM 档案啦！

```
[root@linux SPECS]# rpmbuild -bb showvbird.spec
.....中间省略.....
Requires: /bin/bash
Checking for unpackaged file(s): /usr/lib/rpm/check-files %{buildroot}
Wrote: /usr/src/redhat/RPMS/i386/showvbird-1.0-1.i386.rpm
```

最后这个被打包成功的档案就被放置在 /usr/src/redhat/RPMS/i386/showvbird-1.0-1.i386.rpm 啰！然后给他安装一下：

```
[root@linux SPECS]# rpm -ivh ../RPMS/i386/showvbird-1.0-1.i386.rpm
Preparing...                ########################################### [100%]
   1:showvbird              ########################################### [100%]

[root@linux SPECS]# rpm -qi showvbird
Name        : showvbird                 Relocations: (not relocatable)
Version     : 1.0                            Vendor: (none)
Release     : 1                          Build Date: Mon Oct  3 11:08:30 2005
Install Date: Mon Oct  3 11:11:30 2005   Build Host: linux.site.tw
Group       : VBird's Home               Source RPM: showvbird-1.0-1.src.rpm
Size        : 143                           License: GPL
Signature   : (none)
Packager    : VBird
URL         : http://linux.vbird.org
Summary     : This is a demo RPM package.
```

```
Description :
This package is just a demo RPM.

[root@linux SPECS]# which showvbird.sh
/usr/local/bin/showvbird.sh
[root@linux SPECS]# rpm -ql showvbird
/usr/local/bin/showvbird.sh  <==果然记录起来了！自己的软件呢！
```

用很简单的方式，就可以将自己的软件或者程序给他修改与设定妥当！很不错吧！ 以后您就可以自行设定你的 RPM 啰！当然，也可以手动修改您的 SRPM 的来源档内容啰！

要选择 RPM 还是 Tarball？

- 优先选择 RPM：

这一直是个有趣的问题：『如果我要升级的话，或者是全新安装一个新的套件， 那么该选择 RPM 还是 Tarball 来安装呢？』！基本上，如果有 RPM 可以提供给您的 distribution 来安装，并且没有严重的相依属性的问题时，呵呵！选择 RPM 来安装会是一个比较好的解决方案， Why ？这是由于刚刚上面就提到的 RPM 的好处 啦！可以具有档案与数据均有纪录的优点，这就是上面提到的 /var/lib/rpm 这个目录里面的数据库，这个记录可以让你在管理上更为便利，包括上面提到的 RPM 的升级、安装、验证与移除等等。尤其是在查询上面！可以让你在管理你的系统上面更为便利。

但是 RPM 也不是没有缺点的，包括最为大家所抱怨连连的『 属性相依 』的问题，每一个不同版本之间，就必须要以不同的 RPM 档案来安装！此外，如果要升级『某一个套件』而已时， 通常还需要连带其它的套件也必须要一起升级才行，否则会有问题！此外，当一个套件经过了『 大幅度的修改 』之后，通常旧的 RPM 与新的 RPM 之间已经几乎无法『完全兼容』时， 呵呵！那么升级或者是移除的手续可是会累坏人的！

例如前两年朋友们常常问到的 Apache 1.3.xx 与 2.0.xx 的版本升级问题！ 由于这两个版本之间的架构差异性太大，加上版本属性相依问题，所以很难得到一个完满的解决方案，这个时候 RPM 就不那么合适了。(除非您要一个一个的将 Apache 移除，连同其相依的套件，然后再将 Apache 一个一个的安装，包括新套件的相依套件！ ^_^ .....鸟哥是不会这么做的啦！)

- 简易方法：

如果 RPM 档案并不是这么容易取得的话，这个时候 Tarball 的方式就特别适合您的安装了！这是因为 Tarball 可以自行设定编译时的参数，此外，也可以自行设定『安装路径』， 相当的适合于想要安装『多个不同版本的同一个套件』的情况(说穿了就是测试机器)！

这是怎么说呢？！由于 RPM 必须要配合系统里面其它的相依属性的套件，所以基本上， 他的安装路径(就是每个档案的放置路径)理论上是必须要放在固定的目录的，就是不能随意的改变他的安装路径。 因此，当有两个不同版本的相同套件想要测试的时候，大概一定就得将原先的版本移除之后， 才能安装使用新的版本啰！(此外，由于相依的套件几乎都已经包含在 tarball 当中了，所以安装上面其实并不难啦！)

相对于 RPM 的制式格式， tarball 可就灵活多了！你可以自行编译套件并且将他安装在不同的路径， 只要在启动的时候选择正确的版本，那么不同版本的套件可以同时的存在于一个系统当中， 而且可以透过选

择启动的档案来启动不同的版本。当然啰！你也可以让 tarball 的安装与 RPM 的安装同时存在于一个系统当中，但是需要特别留意的是， 你在启动该套件的时候，千万记得你的启动路径！免得启动到了错误的版本了！呵呵！(这也是一个系统存在不同多个版本的套件容易发生的错误！ 希望大家都能够了解这个问题呢！)

所以说，为了避免这种路径上的错误困扰，基本上，我们都希望 Tarball 的安装路径可以设定在 Linux 原本就规划要给大家安装的路径『 /usr/local 』这个路径下！这样可以省去相当多寻找档案的时间！ 而且在管理上面也会比较容易！呵呵！

不过， Tarball 最麻烦的地方有几点：

- 反安装：
  Tarball 最麻烦的地方就在于他的『解安装』了！ 相当的讨厌！如果是简单的直接将所有的套件安装在一个目录下的话，例如 /usr/local/mrtg 时，那么解安装还算简单，就是将该路径杀掉就 OK 啦！但是如果是类似 sendmail 这一种呢？他的路径都是已经放置死的(需要在 /etc/sendmail.cf、/etc/mail 底下)那么追踪反安装的路径就很烦人；
- 在线查询：
  如果您的安装路径是在 /usr/local 底下的话，那么执行档会被放置到 /usr/local/bin ，或者是 /usr/local/sbin 底下，参数档会放在 /usr/local/etc 底下，在线查询档案会放在 /usr/local/man 底下，所以在设定上面还有查询上面还算简单(路径设定一下即可！)，不过，如果你是将套件安装在单独的路径下呢？例如 /usr/local/mrtg 底下，那么执行档变成了 /usr/local/mrtg/bin 底下，最麻烦的地方就是 man page (在线查询)放置的地点会变成在 /usr/local/mrtg/man 底下了！糟糕！那么预设的 man page 路径就找不到该说明文件啰！ 这个时候就必须要手动的将该路径加入 /etc/man.conf 这个档案中！而且执行文件放置的路径也没有指定，可以经由 (1)Link 的方式或者 (2)设定 PATH 环境变量的方式将该路径加进去啦！确实是比较麻烦的啦！

所以说，RPM 与 Tarball 各有其优缺点，不过，如果有 RPM 的话，那么优先权还是在于 RPM 安装上面，毕竟管理上比较便利，但是如果套件的架构差异性太大， 或者是无法解决相依属性的问题，那么与其花大把的时间与精力在解决属性相依的问题上，还不如直接以 tarball 来安装，轻松又惬意！

重点回顾

- RPM 的全名是 Red Hat Package Manager，原本是由 Red Hat 公司所发展的，流传甚广；
- RPM 类型的套件中，所含有的套件是经过编译后的 binary file ，所以可以直接安装在使用者端(Client)的系统上，不过，也由于如此，所以 RPM 对于安装者的环境要求相当严格；
- RPM 除了将套件安装至使用者的系统上之外，还会将该套件的版本、名称、档案与目录配置、系统需求等等均记录于数据库(/var/lib/rpm)当中，方便未来的查询与升级、移除；
- RPM 可针对不同的硬件等级来加以编译，制作出来的档案可于附档名(i386, i586, i686)来分辨；
- RPM 最大的问题为套件之间的相依性问题；
- SRPM 为 Source RPM ，内含的档案为 Source code 而非为 binary file ，所以安装 SRPM 时还需要经过 compile ，不过，SRPM 最大的优点就是可以让使用者自行修改设定参数(makefile/configure 的参数)，以符合使用者自己的 Linux 环境；

参考资源

刚刚最前面说过了，套件升级最主要的考虑就是『安全性』啦！所以请随时注意安全性方面的问题！ 目前国内的主要安全网站为：『台湾网络危机处理小组』这个组织，请随时注意上面发布的新闻！另外， 如果跟鸟哥一样使用的是 Red Hat 的 distrubution 的话，那么 Red Hat 的 Errata 网页则不可不光临！好啦！底下列出几个 RPM 相关的网页与 Red Hat 的 Errata 网页提供大家参考啰！

- RPM 包装档案管理程序：http://www.study-area.org/tips/rpm.htm
- 中文 RPM HOW-TO：http://www.linux.org.tw/CLDP/RPM-HOWTO.html
- RPM 的使用：http://linux.tnc.edu.tw/techdoc/rpm-howto.htm
- 大家来作 RPM ：http://freebsd.ntu.edu.tw/bsd/4/3/2/29.html
- 一本 RPM 的原文书：http://linux.tnc.edu.tw/techdoc/maximum-rpm/rpmbook/
- Red Hat 的 Errata 网页：http://www.redhat.com/apps/support/errata/

课后练习

- 简单说明 RPM 与 SRPM 的异同?
- 查询系统上的 RPM 套件数据时，系统由何处取得该套件的讯息？(/var/lib/rpm/*)
- 假设我想要安装一个套件，例如 pkgname.i386.rpm ，但却老是发生无法安装的问题，请问我可以加入哪些参数来强制安装他？
- 承上题，您认为强制安装之后，该套件是否可以正常执行？为什么？
- 有些人使用 OpenLinux 3.1 Server 安装在自己的 P-166 MMX ，却发现无法安装， 在查询了该原版光盘的内容，发现里面的文件名称为 ***.i686.rpm 。请问，无法安装的可能原因为何？
- 请问我使用 rpm -Fvh *.rpm 及 rpm -Uvh *.rpm 来升级时，，两者有何不同？

认识系统服务 daemons

最近更新日期：2005/10/12

在 Unix-Like 的系统中，常常听到这个字眼： daemons ！那么什么是传说中的 daemons 呢？这些 daemon 放在什么地方？他的功能是什么？该如何启动这些 daemons ？又如何有效的将这些 daemon 管理妥当！？此外，要如何视察这些 daemons 开了多少个 ports ？又这些 ports 要如何关闭？还有还有，晓得你的系统的这些 port 各代表的是什么服务吗？ 这些都是最基础需要注意的呢！尤其是在架设网站之前，这里个观念就显的更重要了。

什么是 daemon 与服务 (service)

如果您常常上网去查看一些数据的话，尤其是关于 Unix-Like 的相关操作系统，如 FreeBSD, Unix, Linux 等等，应该会常常听到 daemons 这个字眼，那么 daemon 是什么东西呀！？怎么这么常被见到？呵呵，Daemon 的字面上的意思就是『守护神、恶魔？』还真是有点奇怪呦！ ^_^""

先来谈一谈 daemon 这个玩意儿是个啥咚咚？还记得我们在 程序与资源管理 一文当中提到过程序的概念，程序有的在 bash 当中执行程序而触发的，也有开机的时候， 系统自行触发而在背景当中执行的。当然也有系统管理员在开机完成后，登入系统来触发的等等。 不论怎么说，这个 daemon 其实就是一个『在背景当中执行的程序』啦！ 比较特殊的是，所谓的 daemon 通常是负责系统上面的某个服务 (service) ，好让系统可以接受来自使用者或者是网络客户(client)的要求，而加以工作。

那么什么又是服务 (service)？所谓的服务很简单啦，意思是说，主机提供的功能。 这些功能主要分为系统上面的，以及针对网络的服务。针对系统上面的服务，例如我们第四篇提到的 crond 与 atd 等等，他主要负责 Linux 主机上面的工作排程；至于网络服务呢？ 包括远程联机 SSH 服务器，或者是全球信息网 WWW 服务器等等，这些让客户端连接上来取得数据的服务， 就是网络服务啦！

那您了解了，之所以要有主机服务器就是希望他可以提供我们一些网络服务，或者是主机端自己的服务，好

让我们使用者或者是一般用户可以工作的更愉快！而主机要提供这些服务，必须要有相对应的 daemon 来进行服务需求的监听，例如要提供工作排程的服务，就得要有 atd 或者是 crond 这两个 daemon 才行；而 daemon 的启动，其实就是某个程序 (program) 的执行，配合这个程序的设定文件， 就能够有效的启动该程序，加载常驻到内存当中成为 daemon ，并提供相对的服务啰！

一般来说，当我们以 run level 3 或者是 run level 5 完整开机进入 Linux 主机后， 系统已经提供我们很多的服务了！包括打印服务、工作排程服务、邮件管理服务等等； 那么这些服务是如何被启动的？他们的工作型态如何？底下我们就来谈一谈啰！

> Tips:
> 很多时候，我们不会很细的去切分什么是 daemon 而什么是 service， 简单的来说，你可以将 service 与 daemon 视作相同的东西！ 反正就是某个在背景当中执行的程序，他可以提供某些功能就是了！ ^_^



---

daemon 的主要分类

如果依据 daemon 的启动与管理方式，基本上，可以将 daemon 分为可独立启动的 stand alone ， 与透过统一安全机制管理的 Super daemon 两大类，这两类 daemon 的说明是这样的：

- stand_alone：

就字面上的意思来说，stand alone 就是『独立的启动』的意思，也就是说，该 daemon 启动之后，就直接常驻在内存当中啰！他虽然会一直的占用系统的资源， 但最大的优点就是，他会一直启动的啦！所以当有要求来的时候，他就会很快速的响应啰！常常用在这一种 daemon 的网络服务如常见的全球信息网 WWW 的 daemon (httpd) 这一个即是一例！ 因为他需要比较快的响应速度啊！

- super daemon：

相对于 stand alone 的执行方式，这一种服务的启动方式则是藉由统一的一个 daemon 来负责唤起该服务！这一个统一负责的 daemon 就是 inet 这支服务啦！不过，在后来的 Linux 发展套件中，则是使用 xinet 这个设定啰！我们这里以 FC4 的 xinet 来做说明。当有网络的服务要求来的时候，该要求会先送给 xinet 这个服务，然后 xinet 根据该网络要求送来的数据封包的内容 (该内容会记录 IP 与 port) 来将数据封包送给实际运作的服务!而该服务这个时候才会启动的!最常见到的就是 ftp 这支网络服务啦!

这种 daemon 最大的优点就是当没有数据封包来的时候，该服务不会一直占据系统资源 (该服务会在 sleeping 的状态吧！) ，但是相对的，他的反应时间也会比较慢， 因为还要花费一段时间去『唤醒』该服务呀！

那么这两种启动的方式哪一个比较好呢？见仁见智啦！而且还要看该主机的工作负荷与实际的用途说！ 例如当你的主机是用来作为 WWW 服务器的，那么 httpd 自然就以 stand alone 的启动方式较佳！事实上，我们常常开玩笑的说明 stand alone 与 super daemon 的情况，可以银行的窗口来作为说明的范例！

- stand alone ：

在银行里面，假设有一种单一服务的窗口，例如存钱窗口，所以，当你需要存钱的时候， 直接前往该窗口，就有『专人』为您服务啦！

- super daemon ：

在银行里面假设还有另外一种复合型态的窗口，同时提供转帐、资金调度、 提款等等的业务，那当你需要其中一项业务的时候，就需要前往该窗口， 但是坐在窗口的这个营业员，拿到你的需求单之后，往后面一丢『喂！那个转帐的仁兄！该你的工作了』 那么那个仁兄就开始工作去！然而里头还有资金调度与提款等负责业务的仁兄呢？他们在干嘛？ 嘿嘿！看看报、喝喝茶啰！

那么这里就会引出另外一个问题啦！假设银行今天的人潮特别的汹涌， 所以这个窗口后面除了你之外还有很多的人！那么想一想，这个窗口是要『一个完成再来下一个』 还是『全部都把你们的单据拿来，我全部处理掉』呢？呵呵！是不是不太一样？基本上， 针对这种 super daemon 的处理模式有两种，分别是这样：

- multi-threaded：
  就是我们提到的，全部的客户之要求都给他拿来，一次给他交办下去，所以一个服务同时会负责好几个程序。
- single-threaded：
  这个就是目前我们『人类的银行』最常见的方式啦，不论如何，反正一个一个来， 第一个没有处理完之前，后面的请排队！嘿嘿！所以如果 client 的要求突然大增的话，那么这些晚到的 client 可得等上一等！

另外，需要注意的是，既然银行里头有这两种窗口同时存在，所以啰，在 Linux 系统里面，这两种 daemon 是可以同时存在的啦！也就是说，某些服务可以使用 stand alone 来启动，而有其它的服务则可以使用 xinet （ 或者是 inet ） 大致的情况就是这样啦！瞭乎！？

不过，如果以 daemon 的工作状态来区分，则主要分为两类：

- signal-control
  这种 daemon 是透过讯号来管理的，只要有任何需求进来，他就会立即启动去处理！ 例如打印机的服务 (cupsd)
- interval-control
  这种 daemon 则主要是『每隔一段时间就主动的去执行某项工作』， 所以，即使你设定好设定档之后，他也不会立刻执行，而是某个时间点才会去工作。举例来说， atd 与 crond 就是这种 (每分钟执行一次！)

另外，如果您对于开发程序很有兴趣的话，那么可以自行查阅一下『 man 3 daemon 』 看看系统对于 daemon 的详细说明吧！ ^_^。

与服务有关的端口口对应资料： /etc/services

现在我们知道系统所提供的服务是执行某个 program ，由该程序的功能所提供的。 也知道一部主机上面可能会同时拥有多个服务，当然，可能会有多个网络服务同时存在。 此时你会不会觉得很奇怪啊？我一部主机同时开启 WWW 与 FTP 时， 客户端跟我要数据，那么主机会响应什么数据给客户端啊？奇不奇怪呢？

其实，就如同上面提到的人类的银行一样，不同的服务有不同的窗口号码， 同样的，在 Linux 系统上面，不同的网络服务，确实有不一样的监听埠口 (listen port)。 我们可以透过指定指向主机的某个端口口 (port) 来连上我们想要的服务呢！ 举例来说，我们可以在浏览器上面输入这样的网址：

- http://ftp.isu.edu.tw/
- ftp://ftp.isu.edu.tw/

有没有发现，两个网址都是指向 ftp.isu.edu.tw 这个义守大学的 FTP 网站， 但是浏览器上面显示的结果却是不一样的？是啊！这是因为我们指向不同的服务嘛！ 一个是 http 这个 WWW 的服务，一个则是 ftp 这个服务，当然显示的结果就不同了。

那我们怎么知道那个 port 是由那个服务所启动的呢？ 因为目前已经有很多既定的网络通讯协议，这些通讯协议使用的 port 是固定的， 也是公认的标准的 port number ，我们可以称为 well known 的信息。 那么我们 Linux 主机有没有相关的信息呢？当然有啊！那就是 /etc/services 这个档案啊！我们取 FC4 的这个档案一部份来说明：

```
[root@linux ~]# vi /etc/services
......省略......
ftp-data        20/tcp
ftp-data        20/udp
ftp             21/tcp
ftp             21/udp          fsp fspd
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
telnet          23/udp
......省略......
http            80/tcp          www www-http    # WorldWideWeb HTTP
http            80/udp          www www-http    # HyperT
pop3            110/tcp         pop-3           # POP version 3
pop3            110/udp         pop-3
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper TCP
sunrpc          111/udp         portmapper      # RPC 4.0 portmapper UDP
netbios-ns      137/tcp                         # NETBIOS Name Service
netbios-ns      137/udp
netbios-dgm     138/tcp                         # NETBIOS Datagram Service
netbios-dgm     138/udp
netbios-ssn     139/tcp                         # NETBIOS session service
netbios-ssn     139/udp
......省略......
# 这个档案的内容是以底下的方式来编排的:
# <daemon name>   <port 与数据型态>   <该服务的说明>
```

像上面说的是，第一栏为 daemon 的名称、第二栏为该 daemon 所使用的 port 号码与其网络数据封包传送时候的类型，主要为确定联机后才进行数据传输的可靠的 TCP 封包， 以及较快速但不确定性较高的 UDP 封包等。 举个例子说，那个 e-mail 的发信协议为 smtp 这个服务，而这个服务的使用之 port 即为 25 啦！ 就这样！

Tips:
请特别注意！虽然有的时候您可以藉由修改 /etc/services 来更改一个服务的 port 号，不过并不建议如此做，因为很有可能会造成一些协议的错误情况！ 这里特此说明一番呦！(除非您要架设一个地下网站，否则的话，使用 /etc/services 原先的设定就好啦！)



Daemon 的命名规则：

每一个服务的开发者，当初在开发他们的服务时，都有特别的故事啦！ 不过，无论如何，这些服务的名称被建立之后，被挂上 Linux 使用时，通常在服务的名称之后会加上一个 d ，例如例行性命令的建立的 at, 与 cron 这两个服务，通常会被称为 atd 与 crond，这个 d 代表的就是 daemon 的意思。所以，在资源管理那一章中，我们使用了 ps 与 top 来观察程序时，都会发现到很多的 xxxd 的程序，呵呵！通常那就是一些 daemon 的程序啰！

系统的 Daemons 放在哪里：

我们说过， daemon 其实是一支可以在背景执行的程序，这个程序可以负责系统的某个服务。 而既然要负责某个服务，当然啰，就需要有所谓的设定档啰～而为了让使用者可以很轻易的启动该服务， 因此各主要的 Linux distributions 都会替他们的系统进行较有亲和力的启动 daemon 的方式， 那就是利用 shell script 啦！这也是为何我们会在第三篇的时候建议您务必要学习 shell script 的原因啊！^_^

举个例子来说，在 FC4 上面管理系统登录文件的服务为 syslogd 这个 daemon， 那么你如何启动这个 daemon 呢？可以查询一下 man 8 syslogd 来看看到底他需要如何被启动。 想必看的结果是『很烦ㄟ！』干嘛要这样启动啊！真是麻烦～ 此时，启动 syslogd 这个 daemon 的 shell script (/etc/init.d/syslog) 就帮上忙了！ 你只要『 /etc/init.d/syslog restart 』就能够重新启动 syslogd 呢！ 真是很方便啊！而该 shell script 就会主动的去读取相关的设定档，好让我们的设定生效啊！^_^

OK！那么这些 daemons 的 shell scripts 放在哪里啊？他们放置的地方依据 stand alone 与 super daemon 的差异而有所不同，基本上，是放在这些地方：

- stand alone：
  这个放置在 /etc/init.d/ 这个目录里面，几乎所有的 RPM 安装的套件之启动 scripts 都在这里啦！不过，实际上，我们的 FC4 是放置到 /etc/rc.d/init.d/* ， 但你依旧可以记忆成 /etc/init.d ，因为所有的 unix like 机器都有这个目录！
- super daemon：
  这个工作的那一支服务其实就是 xinet 或者是 inet 啦！请注意， xinet 也是一个 daemon 呢！他是 stand alone 启动的，也就是他会一直在监听大家的需求，所以 xinet 的启动 scripts 写在 /etc/init.d/xinetd 这个 scripts 里面啰！但是挂在这个 daemon 里头的服务之设定项目呢？嗯！就是写在 /etc/xinetd.conf 与 /etc/xinetd.d/* 这个目录里面的任何档案！

更详细的来说明每个目录底下的设定的话，总的来说，是这样的：

- /etc/init.d/*

OK！先来了解一下 stand alone 的 daemon 是怎么启动的呢？！很简单，假如我们要启动 syslog 这支记录登录文件的服务，那么要启动他的话，就直接下达：

```
[root@linux ~]# /etc/init.d/syslog start
[root@linux ~]# servcie syslog start
```

那个 service 是一支程序，基本上，也只是用来启动 /etc/init.d/ 底下的 shell script 而已～至于指令或者是档案后面接的参数，亦即是档名之后加上 start 即可，或者是使用 Red Hat 系统有的这个 service script 来进行启动的功能！如果你还记得我们前几节提到过的 shell scripts 的话，那么或许还记得 case ..... esac 这个有选择性的项目的语法吧！？没错！这几支服务就是以 bash scripts 里头的 case 语法写成的！因此，只要加上后面的参数，如此一来， scripts 就会自动的去找寻执行档来执行啰！如果有兴趣的话， 可以在你的系统里面的该目录下开一个档案来观看一下，就知道如何写啰！

- /etc/xinetd.conf

这个档案就是设定 xinet 服务的参数档案啦！

- /etc/xinetd.d/*

这个目录里面的所有档案就是个别挂上 xinet 的所有服务啦！例如赫赫有名的 wu-ftpd 及 telnet 与 pop3 等等！

---

daemon 的启动方式： service

知道了一些有关 daemon 的相关知识后，再来，那么我们如何启动一个 daemon 呢？ 其实，我们知道所谓的 daemon 就是一支可以在系统背景下面运作的程序 (program) 啊， 所以，要启动该 daemon ，就是找到他的执行档，执行他就是了。 不过，因为该 daemon 的执行档所需要加的参数太多了！举例来说，你可以使用『man syslogd』及 『man sshd』来查阅一下该 daemon 要启动时的设定参数！

为了克服这样的困扰，所以各主要 Linux distributions 都会针对该服务设计一个比较亲和的 shell script 来进行启动的程序啊!那就是 /etc/init.d/ 底下的档案,以及 /etc/xinetd.d/ 底下的设定数据。因此，启动服务的方法就变得很简单了。 只要设定好该服务的设定档，然后下达：

```
1. 启动 stand alone 服务的方式：以 syslog 为例：
[root@linux ~]# /etc/init.d/syslog start

2. 启动 super daemon 服务的方式：以 telnet 为例：
[root@linux ~]# vi /etc/xinetd.d/telnet (设定方式参考下节)
[root@linux ~]# /etc/init.d/xinetd restart
```

另外，除了这样的启动方式之外，我们还可以透过 Fedora (Red Hat 系统) 所提供的 service 这个程序来进行 daemon 的启动喔！其实 service 仅是一支 script 啦， 他可以解析后面带有的参数，然后去到 /etc/init.d/ 去启动相对应的服务名称的 script 而已！ 有兴趣的话，可以自行去解析 /sbin/service 这支 shell script 啊！ 底下我们大略说明一下他的用法！

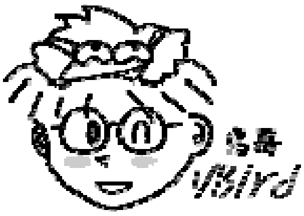
```
[root@linux ~]# service [service name] (start|stop|restart|...)
参数：
service name：亦即是需要启动的服务名称，需与 /etc/init.d/ 对应；
start|...    ：亦即是该服务要进行的工作。
```

```
范例：

范例一：重新启动 crond 这支 daemon ：
[root@linux ~]# service crond restart
[root@linux ~]# /etc/init.d/crond restart
```

在上面的范例当中，其实启动方式以 service 这个程序，或者直接去到 /etc/init.d/ 底下启动， 都一样啦！自行去解析 /sbin/service 就知道为啥了！ ^_^

> Tips:
> 事实上，在 Linux 系统中，要『开或关某个 port 』，就是需要『 启动或关闭某个服务』啦！因此，你可以找出某个 port 对应的服务，程序对应的服务， 进而启动或关闭他，那么那个经由该服务而启动的 port ，自然就会关掉了！



解析 super daemon 的设定档

前面提到，Super daemon 就是一支总管许多服务的 daemon ，这支 daemon 在 FC4 上面即是 xinet 啰～通常我们也称呼为 xinetd 啦～这支 daemon 来管理许多的服务是有好处的， 最大的优势就是『安全性较高！』。怎么说呢？因为 super daemon 可以透过额外的资料分析， 来管理谁可以、谁不能使用某个服务，因此，多了一道类似防火墙的手续，自然就能够比较安全一些啦。 而且他还可以记录该服务的使用状态，也可以记录错误登入的信息，用在管理一些比较危险的服务上面， 确实有他的必要性啦！

底下我们就来谈一谈，这个 super daemon 到底是如何分析的， 当然，就得要先谈一谈，这个 xinetd 的主要预设参数档： /etc/xinetd.conf 啰～

解析 xinetd.conf

先来看一看预设的 /etc/xinetd.conf 这个档案的内容是什么吧！

```
[root@linux ~]# vi /etc/xinetd.conf
#
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/
defaults
{
  instances         = 60       <==同一服务的同时联机数最多可达 60 个
  log_type          = SYSLOG authpriv <==登录后，会被纪录到登录文件的信息
  log_on_success    = HOST PID <==若成功的登入时，记录的信息有哪些？
  log_on_failure    = HOST     <==若登入失败，则记录的信息又是如何？
  cps               = 25 30    <==同一秒钟内最大联机数量为 25 个，若超过 25 个，
                                  则该服务会暂时停止 30 秒！
```

```
}

includedir /etc/xinetd.d <==更多的设定值在 /etc/xinetd.d 那个目录内
```

基本上，这个预设参数档的意义是：『当某个使用 super daemon 管理的服务启动时，除非该服务已经设定好管理的项目，否则将以上述 xinetd.conf 内的预设参数带入。』的意思，也就是说，这仅是默认值，但我们可以自行指定新的设定值来取代 xinetd.conf 内的默认值啦！ 也就是说，这个档案设定成，在预设的状态下『：一个服务最多可达 60 个联机，且同一秒内连接上的联机不可超过 25 个。而若登入的成功与否时， 会分别记录不同的信息到登录文件当中。』这样说，可以比较清楚了吧？ ^_^ 至于更多的参数说明，我们会在底下再强调的！

既然这只是个预设参数档，那么自然有更多的服务参数档案啰～没错～而所有的服务参数档都在 /etc/xinetd.d 里面，这是因为上表当中的最后一行啊！这样瞭了吧！ ^_^。 那么每个参数档案的内容是怎样呢？一般来说，他是这样的：

```
service  <service_name>
{
       <attribute>   <assign_op>   <value>   <value> ...
       ............
}
```

第一行一定都有个 service ，至于那个 <service_name> 里面的内容， 则与 /etc/services 有关，因为他可以对照着 /etc/services 内的名称与 port number 来决定所要启用的 port 是那个啊！然后相关的参数就在两个大刮号中间。 attribute 是一些 xinetd 的管理参数， assign_op 则是参数的设定方法。 assign_op 的主要设定形式为：

= ： 表示后面的设定参数就是这样啦！

+= ： 表示后面的设定为『在原来的设定里头加入新的参数』

-= ： 表示后面的设定为『在原来的参数舍弃这里输入的参数！』

用途不太相同，敬请留意呦！好了！底下再来说一说那些 attribute 与 value ！

attribute (功能)	assing_op (允许的动作)	说明与范例
一般设定项目：		
disable	yes no	允许该 server 可以执行或者是不能执行！当设定为 yes 表示该服务不能执行！ 这个设定是一定要的啦。如果我想要启动某个服务，那么这里就要设定成为： disable = no
socket_type	stream dgram raw	stream 为联机机制较为可靠的 TCP 封包，若为 UDP 封包则使用 dgram 机制。 raw 代表 server 需要与 IP 直接对谈！例如 telnet 使用 TCP ，所以： socket_type = stream
protocol	tcp udp	这个东西说的是，联机的状态使用的是哪一种协议！？各个协议的代号可以参考 /etc/protocols 内容！此外，除非是你自己设定的服务，否则这个可以不用设定

		啦！
wait	yes no	这就是我们刚刚提到的 Multi-threaded 与 single-threaded 的方式啦！一般来说，我们希望大家的要求都可以同时被启用，所以可以设定 wait = no
user	UID root	还记得我们在 账号管理 那一篇提到的 UID 概念吗？对啦！这个 UID 就是那个 UID 啦！要注意的是，假如你的服务启动者不要以 root 为主的话，那么这个地方就可以改变其它的使用者，例如 nobody ！这个咚咚也会有安全防护的机制存在！此外，需要注意这个 UID 必须存在于 /etc/passwd 。
group	GID	跟 user 的意思相同！只是这个 GID 的使用者也必须存在于 /etc/group 当中！
instances	number UNLIMITED	这个是『在同一时间之内，同一个服务可以允许的联机数目』的意思， 你可以写入一个『数字』来控制联机数目，也可以使用 UNLIMITED 来告诉系统『没有上限』啰！例如你在同时段之内仅允许 ftp 联机有 30 个，那么这里就可以输入 30 啦！
nice	-19 ~ 19	还记得我们在 程序管理 里面谈到的那个 nice 指令吗？！对啦！这里就是这个东西啰！数字越小(负值)代表该程序越优先被执行！
server	program 完整檔名	这个就是指出这个服务的启动程序！例如要启动 telnet 的话，其实就是 in.telnetd 这支程序啦！所以这个时候在这里输入 server = /usr/sbin/in.telnetd
server_args	program 一些参数	这里应该输入的就是你的 server 那里需要输入的一些参数啦！例如 in.telnetd 当中，我们还可以加入某些参数！
log_on_success	PID HOST USERID EXIT DURATION	在『成功登入』之后，需要记录的项目：PID 为纪录该 server 启动时候的 process ID ， HOST 为远程主机的 IP、USERID 为登入者的账号、EXTI 为离开的时候记录的项目、 DURATION 为该使用者使用此服务多久？
log_on_failure	HOST USERID ATTEMPT RECORD	当登入失败之后被 syslog 登入的项目:HOST 为远程主机的 IP，USERID 为登入者账号、 ATTEMPT 为记录登入失败者企图的意图为何、RECORD 为记录远程主机的信息！以及为何本机 server 不能启动的原因！主要有 login, shell, exec, finger 等指令可以使用在这里！(基本上，可以在 /etc/hosts.allow 或 /etc/hosts.deny 书写内容)。

进阶设定项目：		
env	'name=value'	这一个项目可以让你设定环境变量，环境变量的设定规则可以参考 认识 BASH Shell 。
port	number	这里可以设定不同的服务与对应的 port ，但是请记住你的 port 与服务名称必须与 /etc/services 内记载的相同才行！
redirect	IP_Address port	将 client 端对我们 server 的要求，转到另一部主机上去！呵呵！这个好玩呦！ 例如当有人要使用你的 ftp 时，你可以将他转到另一部机器上面去！ 那个 IP_Address 就代表另一部远程主机的 IP 啰！
includedir	directory	表示将某个目录底下的所有档案都给他塞进来 xinetd.conf 这个设定里头！这东西有用多了， 如此一来我们可以一个一个设定不同的项目！而不需要将所有的服务都写在 xinetd.conf 当中！ 你可以在 /etc/xinetd.conf 发现这个设定呦！
安全控管项目：		
bind	IP_Address	这个是设定『允许使用此一服务的适配卡』的意思！举个例子来说，你的 Linux 主机上面有两个 IP ，而你只想要让 IP1 可以使用此一服务，但 IP2 不能使用此服务，这里就可以将 IP1 写入即可！那么 IP2 就不可以使用此一 server 啰
interface	IP_Address	与 bind 相同
only_from	0.0.0.0 192.168.1.0/24 host_name domain_name	这东西用在安全机制上面，也就是管制『只有这里面规定的 IP 或者是主机名称可以登入！』如果是 0.0.0.0 表示所有的 PC 皆可登入，如果是 192.168.1.0/24 则表示为 C class 的网域！亦即由 192.168.1.1 ~ 192.168.1.255 皆可登入！另外，也可以选择 domain name ，例如 .ev.ncku.edu.tw 就可以允许成大环工系的网域 IP 登入你的主机使用该 server ！
no_access	0.0.0.0 192.168.1.0/24 host_name domain_name	跟 only_from 差不多啦！就是用来管理可否进入你的 Linux 主机启用你的 server 服务的管理项目！ no_access 表示『不可登入』的 PC 啰！
access_times	00:00-12:00 HH:MM-HH:MM	这个项目在设定『该服务 server 启动的时间』，使用的是 24 小时的设定！例如你的 ftp 要在 8 点到 16 点开放的话，就是： 08:00-16:00。
umask	000 777 022	还记得在 档案权限 里面约略提过的 umask 这个东西吗？呵呵！没错！就是那个鬼玩意儿啰！ 可以设定使用者建立目录或者是档案时候的属性！系统建议值是 022 。

OK!我们就利用上面这些参数来架构出我们所需要的一些服务的设定吧！参考看看底下的设定方法啰！ ^_^

一个简单的 telnet 范例设定

我们说过，使用 super daemon 来管理主机，最大的优点就是多了一道管理的手续， 所以，可以进行比较多的监控动作，像上一个小节我们提到的相关参数当中， 就能够发现到一些端倪了。在这里，我们举个简单的例子来说明一下整个 super daemon 的管理吧！但是要设定 telnet 的话，就得要安装 telnet 才行。在 FC4 的版本上，我们安装的是 telnet-server-0.17-35 这个套件资料， 请您先以 rpm 的方式 来安装喔！ ^_^

在预设的 /etc/xinetd.d/telnet 内容是这样的：

```
[root@linux ~]# vi /etc/xinetd.d/telnet
service telnet
{
        flags           = REUSE   <==额外的参数使用 REUSE
        socket_type     = stream  <==使用 TCP 的封包格式
        wait            = no      <==可以有多个联机同时连进来
        user            = root    <==启动者预设为 root
        server          = /usr/sbin/in.telnetd <==使用的是这支程序!
        log_on_failure  += USERID <==若登入错误, 『加计』记录使用者 ID
        disable         = yes     <==此服务预设关闭!
}
```

其实，主要的参数可以参考上一小节的表格，也可以直接利用『 man xinetd.conf 』来查阅！ 不过，如果你对于这样的设定并不满意的话，其实还可以手动来修改呢！ 因为我们知道， telnet 并不是个十分安全的服务，详细机制可以参考 服务器篇 的 远程联机服务器 来查阅， 所以，如果你想要更多的安全机制，举例来说，你想要让 telnet 在局域网络内与 Internet 上面的联机机制有差异时，例如这样：

- 对内部网域开放较多权限的部分：
  假设 Linux 主机有两张网络卡，对内的这一张 IP 为 192.168.1.100 ，且仅针对 192.168.1.0/24 这个网段提供登入。然后开放所有与 telnet 有关的权限， 包含总联机数量与联机时间等。但是， 192.168.1.120 及 192.168.1.130 两个 IP 不允许登入；
- 对外部网域较多限制的设定：
  对外的 IP 假设为 140.116.44.125 ，且仅允许台南的校园网络 (140.116.0.0/16)， 以及教育界的主机名称 (.edu.tw)，另外，仅开放早上 1~9 点及 20~24 两个时段登入而已。 此外，最多容许十个联机进入。

在这样的规划情况下，我可以将刚刚上头的 /etc/xinetd.d/telnet 这个档案修改成为：

```
[root@linux ~]# vi /etc/xinetd.d/telnet
# 先针对对内的较为松散的限制来设定：
service telnet
{
```

```
    disable         = no                    <==预设就是启动 telnet 服务
    bind            = 192.168.1.100         <==只允许经由这个适配卡的封包进来
    only_from       = 192.168.1.0/24        <==只允许 192.168.0.0/24 这个网段
                                               的主机联机进来使用 telnet 的服务
    no_access       = 192.168.1.{120,130}   <==不许这些 PC 登入
    instances       = UNLIMITED             <==同时允许联机不限制!
    nice            = 0                     <==使用的优先级较高
    flags           = REUSE                 <==额外使用的参数
    socket_type     = stream                <==使用 tcp 封包常用的联机型态
    wait            = no                    <==不需等待，可以同时允许多个联机
    user            = root                  <==启动程序的使用者身份
    server          = /usr/sbin/in.telnetd<==服务启动的程序
    server_args     = -a none               <==上面那个程序的参数
    log_on_failure  += USERID               <==错误登入时，要记录下来的内容
}

# 再针对外部的联机来进行限制呢!
service telnet
{
    disable         = no                    <==预设就是启动 telnet 服务
    bind            = 140.116.44.125        <==只允许经由这个适配卡的封包进来
    only_from       = 140.116.0.0/16        <==只允许 140.116.0.0 ~ 140.116.255.255
                                               这个网段联机进来使用 telnet 的服务
    only_from      += .edu.tw               <==累加设定，只有教务界才能联机!
    access_times    = 1:00-9:00 20:00-23:59
                                            <==每天只有这两个时段开放服务
    umask           = 022                   <==建立档案时的预设属性设定
    instances       = 10                    <==同时只允许 10 个联机
    nice            = 10                    <==使用的优先级较低
    flags           = REUSE                 <==额外使用的参数
    socket_type     = stream                <==使用 tcp 封包常用的联机型态
    wait            = no                    <==不需等待，可以同时允许多个联机
    user            = root                  <==启动程序的使用者身份
    server          = /usr/sbin/in.telnetd<==服务启动的程序
    server_args     = -a none               <==上面那个程序的参数
    log_on_failure  += USERID               <==错误登入时，要记录下来的内容
}
```

在上面这个范例当中，我们用了很多的网络 IP 显示方式，包括 192.168.1.0/24 ， 以及 140.116.0.0/16 ，这代表『 192.168.1.0~192.168.1.255 的所有 IP 』以及 『 140.116.0.0 ~140.116.255.255 所有的 IP 』更详细的说明，我们会在服务器篇内详谈的。 用了这个设定值之后，你会发现你的 telnet 针对两个网段来设计了！ 设计完成之后，由于这是 xinetd 的设定档，所以启动的方式与观察的方式为：

```
# 如果您的 telnet 本来就有启动的话，那么会发现有一个联机存在你的系统中
```

```
[root@linux ~]# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address    Foreign Address   State    PID/Program name
tcp        0      0 0.0.0.0:23       0.0.0.0:*         LISTEN   19255/xinetd
# 看到喔！是 xinetd 的 program name 呢！

# 重新修改 /etc/xinetd.d/telnet 之后，重新启动的方式与观察为：
[root@linux ~]# /etc/init.d/xinetd restart
[root@linux ~]# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address       Foreign Address   State    PID/Program name
tcp        0      0 140.116.44.125:23   0.0.0.0:*         LISTEN   19281/xinetd
tcp        0      0 192.168.1.100:23    0.0.0.0:*         LISTEN   19281/xinetd
# 有没有看到两个接口啊～而且， PID 会是同一个呢！
```

呵呵！如上面的设定，我们可以将 telnet 的启动项目进行更多的限制！ 如此一来，将有助于我们的安全防护呢！尤其如果可以针对不同的接口来设定，嘿嘿！ 就更加的棒啰！不过，请注意喔！如果照上面的设定，那么您的主机上面将会开了两个 23 port 的接口，分别是给两个接口来使用的呢！嗯！真好玩?同样的， 你也可以针对自己的喜好来设定你的其它 daemon 使他挂在 xinetd 底下呢！

TCP_Wrappers

事实上，除了使用 xinetd 的设定档来设定安全机制之外， 我们还可以利用额外的机制来抵挡某些不受欢迎的资料来源喔！ 那就是 /etc/hosts.allow 以及 /etc/hosts.deny 这两个档案的功能啦！ 这两个档案可以藉由分析：

- 启动的服务名称 (daemon 执行档档名)；
- 客户端的 IP 来源或网段来源。

来进行客户端使用者是否能够登入的判断呢！不过，虽然这两个档案已经被整合到 xinetd 里面去了， 不过，要获得更多的功能，还是得要安装 tcp_wrappers 这个套件才行！ 因为，这两个档案本身就是 tcp_wrappers (其实是 /usr/sbin/tcpd 那个档案而已啦！) 的设定档啊！ 而他也可以整合到整个系统的服务里头去，可以算是最最基础的一个防火墙架构啦！ ^_^

其实， /etc/hosts.allow 与 /etc/hosts.deny 是 /usr/sbin/tcpd 的设定档， 而这个 /usr/bin/tcpd 则是用来分析进入系统的 TCP 封包的一个软件，他是由 TCP Wrappers 所提供的。 那为什么叫做 TCP_Wrappers 呢？那么 wrappers 有包裹的意思，所以说，这个套件本身的功能就是在分析 TCP 网络数据封包啦！那么刚刚我们稍微提到我们网络的封包数据主要是以 TCP 封包为主，这个 TCP 封包的文件头至少记录了来源与目主机的 IP 与 port ，因此，若藉由分析 TCP 封包，就可以比对看我要不要让这个数据进入到主机里面来啰！所以啦，我们要使用 TCP_Wrappers 来控管的，就是：

1. 来源 IP
2. port (就是服务啦)

TCP_Wrappers 设定 TCP 封包是否可以进入的设定档在 /etc/hosts.allow 与 /etc/hosts.deny 当中。因此，基本上，如果一个服务是受到 xinetd 或 TCP_Wrappers 的控制时，那么该服务就会受限于 hosts.allow 与 hosts.deny 的管理了！而如果你自己安装的套件当中（ 亦即使用 Tarball 安装的方式之套件 ），除非有自行定义支持 TCP_Wrappers 的功能 ，否则就无法使用这个玩意啰！嘿嘿！

那么这两个档案是干嘛用的？刚刚不是提过哪！他主要是用来规范 TCP 封包的规则的，所以呢， 里面记录的当然就是：『某些 IP 在特定服务中是否能够进入主机』！那么要怎么写？ 这两个档案的内容基本的语法是：

```
<service(program_name)> : <IP, domain, hostname> : <action>
```

所以我们要先找出来那个 service_name 才行，例如以我们刚刚的 telnet 为例，那个 service_name 是什么呢？其实指的就是在 xinetd.conf 设定档中的 server 这个设定后面接的程序名称啦！所以，telnet 在 FC4 底下的名称为 in.telnetd 因此，如果你不想让 140.116.44.202 这个地址及 140.116.32.0/255.255.255.0 这个 C class 的网域进入你的主机的话，那么可以这样在 /etc/hosts.deny 里面设定：（ 关于 IP，网域，网段，还有相关的网络知识，在这个基础篇当中我们不会谈到， 详细的数据请先自行参考服务器架设篇的内容！ ）

```
[root@linux ~]# vi /etc/hosts.deny
in.telnetd : 140.116.44.202 140.116.32.0/255.255.255.0 : deny
```

当然也可以写成两行，亦即是：

```
[root@linux ~]# vi /etc/hosts.deny
in.telnetd : 140.116.44.202              : deny
in.telnetd : 140.116.32.0/255.255.255.0 : deny
```

这样一来，对方就无法以 telnet 进入你的主机啦！方便吧！不过，既然如此，为什么要设定成 /etc/hosts.allow 及 /etc/hosts.deny 两个档案呢？其实只要有一个档案存在就够了， 不过，为了设定方便起见，我们存在两个档案，其中需要注意的是：

- 写在 hosts.allow 当中的 IP 与网段，为预设『可通行』的意思，亦即最后一个字段 allow 可以不用写；
- 而写在 hosts.deny 当中的 IP 与网段则预设为 deny ，第三栏的 deny 亦可省略；
- 这两个档案的判断依据是： (1) 以 /etc/hosts.allow 为优先，而 (2) 若分析到的 IP 或网段并没有纪录在 /etc/hosts.allow ，则以 /etc/hosts.deny 来判断。

也就是说，/etc/hosts.allow 的设定优先于 /etc/hosts.deny 啰！了解了吗？基本上，只要 hosts.allow 也就够了，因为我们可以将 allow 与 deny 都写在同一个档案内， 只是这样一来似乎显得有点杂乱无章，因此，通常我们都是：

1. 允许进入的写在 /etc/hosts.allow 当中；
2. 不许进入的则写在 /etc/hosts.deny 当中。

此外，我们还可以使用一些特殊参数在第一及第二个字段喔！内容有：

- ALL：代表全部的 program_name 或者是 IP 都接受的意思，例如 ALL: ALL: deny
- LOCAL：代表来自本机的意思，例如： ALL: LOCAL: allow

- UNKNOWN：代表不知道的 IP 或者是 domain 或者是服务时；
- KNOWN：代表为可解析的 IP, domain 等等信息时；

再强调一次，那个 service_name 其实是启动该服务的程序，举例来说，/etc/init.d/ssh 这个 script 里面， 实际上启动 ssh 服务的是 sshd 这个程序，所以，你的 service_name 自然就是 sshd 啰！ 而 /etc/xinetd.d/telnet 内有个 server 的设定项目， 那个项目指到 in.telnetd 这个程序来启动的喔！要注意的很！(请分别使用 vi 进这两支 scripts 查阅) 好了，我们还是以 telnet 为例子来说明好了，现在假设一个比较安全的流程来设定，就是：

1. 只允许 140.116.44.0/255.255.255.0 与 140.116.79.0/255.255.255.0 这两个网域，及 140.116.141.99 这个主机可以进入我们的 telnet 服务器；
2. 此外，其它的 IP 全部都挡掉！

这样的话，我可以这样设定：

```
[root@linux ~]# vi /etc/hosts.allow
in.telnetd:  140.116.44.0/255.255.255.0
in.telnetd:  140.116.79.0/255.255.255.0
in.telnetd:  140.116.141.99
in.telnetd:  LOCAL

[root@linux ~]# vi /etc/hosts.deny
in.telnetd: ALL
```

那么有没有更安全的设定，例如，当当有其它人扫瞄我的 telnet port 时，我就将他的 IP 记住！以做为未来的查询与认证之用！ 是有的！只是，那就得要有额外的动作参数加在第三栏了。主要的动作有：

- spawn (action)
  可以利用后续接的 shell 来进行额外的工作，且具有变量功能，主要的变量内容为： %h (hostname), %a (address), %d (daemon)等等；
- twist (action)
  立刻以后续的指令进行，且执行完后终止该次联机的要求 (DENY)

我们知道 finger 可以反向追踪网络封包的来源，所以，我希望这样：

1. 利用 safe_finger 去追踪出对方主机的信息；
2. 将该追踪到的结果以 email 的方式寄给 root ；
3. 在对方屏幕上面显示不可登入的讯息

此时可以利用 spwan (action1) | (action2) : twist (action3) 来进行， 也就是说，其实在 /etc/hosts.deny 的第三个字段可以继续延伸下去的！整个信息有如这样：

```
[root@linux ~]# vi /etc/hosts.deny
in.telnetd: ALL: spawn (echo "security notice from host `/bin/hostname`" ;\
          echo; /usr/sbin/safe_finger @%h ) | \
          /bin/mail -s "%d-%h security" root & \
```

```
        : twist ( /bin/echo -e "\n\nWARNING connection not allowed.\n\n" )
```

在上面的例子中，第三行的 root 那个账号，可以写成你的个人账号或者其它 e-mail ，以免很少以 root 身份登入 Linux 主机时，容易造成不知道的情况，另外，最后几行，亦即 :twist 之后的那几行为同一行。如此一来，当未经允许的计算机尝试登入你的主机时， 对方的屏幕上就会显示上面的最后一行，并且将他的 IP 寄到 root (或者是你自己的信箱)那里去！ 另外请注意，那个 /usr/sbin/safe_finger 是由 tcp_wrappers 套件所提供的， 所以您必须要安装该套件才行喔！ ^_^

系统开启的服务

好了，现在假设您已经知道了 daemons 的启动档案放置的目录，也知道了服务与 port 的对应，那么要如何查询目前系统上面已经启动了的服务呢？不要再打混了！已经学过了 ps 与 top 应该要会应用才对耶！呵呵！没错，可以使用 ps 与 top 来找寻已经启动了的服务的程序与他的 PID 呢！不过，我们怎么知道该服务启动的 port 是哪一个？呵呵！好问题！可以直接使用 netstat 这个网络状态观察指令来检查我们的 port 呢！甚至他也可以帮我们找到该 port 的程序呢(PID)！这个指令的相关用途，我们在 程序与资源管理 那一章已经讲过了， 不清楚的话请回去查一查先～这里仅介绍如何使用喔～

观察系统启动的服务：

观察系统已启动的服务方式很多，不过，我们最常使用 netstat 来观察。 基本上，以 ps 来观察整个系统上面的服务是比较妥当的，因为他可以将全部的 process 都找出来。 不过，我们比较关心的，还是在于有启动网络监听的服务啊，所以， 鸟哥会比较喜欢使用 netstat 来查阅啦。

```
范例一：找出目前系统开启的『网络服务』有哪些?
[root@linux ~]# netstat -tulp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address State  PID/Program name
tcp        0      0 *:ftp           *:*             LISTEN 1605/vsftpd
tcp        0      0 *:pop3          *:*             LISTEN 1613/dovecot
tcp        0      0 *:ssh           *:*             LISTEN 1587/sshd
udp        0      0 *:bootpc        *:*                    26035/dhclient
# 看一看上头，Local Address 的地方会出现主机名称与服务名称，
# 要记得的是，可以加上 -n 来显示 port number ，而服务名称与 port
# 对应则是写在 /etc/services 里头喔！

范例二：找出所有的有监听网络的服务 (包含 socket 状态)：
[root@linux ~]# netstat -lnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address State  PID/Program name
tcp        0      0 *:ftp           *:*             LISTEN 1605/vsftpd
tcp        0      0 *:pop3          *:*             LISTEN 1613/dovecot
tcp        0      0 *:ssh           *:*             LISTEN 1587/sshd
udp        0      0 *:bootpc        *:*                    26035/dhclient
```

```
Active UNIX domain sockets (only servers)
Proto RefCnt Flags     Type   State      I-Node PID/Program name   Path
unix  2       [ ACC ]  STREAM LISTENING 5188   1673/master        private/tlsmgr
unix  2       [ ACC ]  STREAM LISTENING 5192   1673/master        private/rewrite
......以下省略......
# 仔细的瞧一瞧啊，除了原有的网络监听 port 之外，还会有 socket 显示在上面，
# 我们可以清楚的知道有哪些服务被启动呢！

范例三：观察所有的网络连接状态，查询是否有异常的联机。
[root@linux ~]# netstat -anp
# 利用这个指令可以查出有问题的联机，还可取得 PID，
# 可以用来 kill 掉任何一个觉得怀疑的程序呢！
```

利用 netstat 可以取得很多跟网络有关的服务信息，透过这个指令，我们可以轻易的了解到网络的状态，并且可以透过 PID 与 kill 的相关功能，将有问题的数据给他剔除说～ 当然啦，要更详细的取得 PPID 的话，才能够完全的抵挡有问题的程序啦！

另外，除了已经存在系统当中的 daemon 之外，如何在一开机就完整的启动我们所需要的服务呢？ 底下我们就来谈一谈 chkconfig 及 ntsysv 这两个好用的东西！

设定开机后立即启动服务的方法：

就如同上面提到的，我们使用 netstat 仅能观察到目前已经存在于系统当中的 daemon ，使用 service 这个指令或者是 /etc/init.d/* start 的方法， 仅能在目前的环境下启动某个服务而已。那么重新开机后呢？该服务是否还是继续的自动启动？ 这个时候我们就得要复习一下，到底我的 Linux 主机是怎么开机的呢？

1. BIOS
2. MBR (boot loader)
3. kernel loading
4. init program
5. initial script (/etc/rc.d/rc.sysinit)
6. daemon start (/etc/rc.d/rc[0-6].d/*)
7. local setting (/etc/rc.d/rc.local)

大致的情况是这样，看到啰～整个服务可以被开机就启动的地方有两个， 一个是在 daemon start (/etc/rc.d/rc[0-6].d/*) 那个目录下，该目录下的档案主要以 S 及 K 开头，分别代表开机时启动与关机时关闭的意思，更多信息可以参考 开机关机流程与 loader 那个章节～ 也就是说，如果我可以将要启动的服务写入 /etc/rc.d/rc[0-6].d 目录内， 那么该服务就可以在开机的时候自动的被启动了！就是这样简单～

至于另一个也可以在开机时启动的档案，那就是 /etc/rc.d/rc.local 这个档案喔！ 你可以将任何想要在开机时启动的程序写入到这个档案当中，这个档案是以 shell script 的语法写成的，所以你可以轻易的就设定好你想要启动的数据了！ ^_^

好了，既然如此的话，那么我是否要使用 ln 去到 /etc/rc.d/rc[0-6].d 当中设定相关的服务呢？ 不需要的，因为我们 Fedora 有提供两个好用的指令来达成这个功能啊！那就是 chkconfig 与 ntsysv 喔！

- chkconfig

```
[root@linux ~]# chkconfig --list
[root@linux ~]# chkconfig [--add|--del] [service_name]
[root@linux ~]# chkconfig --level [0123456] [service_name] [on|off]
参数：
--list ：仅将目前的各项服务状态列出来
--add  ：增加一个服务名称给 chkconfig 来管理，该 service_name 必须在
         /etc/init.d/ 内！
--del  ：删除一个给 chkconfig 管理的服务
--level：设定某个服务在该 level 下启动 (on) 或关闭 (off)
范例：

范例一：列出目前系统上面所有被 chkconfig 管理的服务
[root@linux ~]# chkconfig --list |more
NetworkManager  0:off   1:off   2:off   3:off   4:off   5:off   6:off
.....中间省略.....
snmpd           0:off   1:off   2:off   3:off   4:off   5:off   6:off
yum             0:off   1:off   2:off   3:off   4:off   5:off   6:off

xinetd based services:
        chargen:        off
.....中间省略.....
        telnet:         off
# 这个 chkconfig 的输出主要分为两大部分，分别是 stand alone 的服务，
# 他会分出 0~6 个 run level 的资料，亦即上半部的显示，至于下半部则是
# super daemon 管理的服务的输出情况！由 super daemon 管理的服务，
# 是没有 run level 之分的喔！

范例二：显示出目前在 run level 3 为启动的服务
[root@linux ~]# chkconfig --list | grep '3:on'

范例三：让 atd 这个服务在 run level 为 3, 4, 5 时启动：
[root@linux ~]# chkconfig --level 345 atd on
```

瞧！ chkconfig 是否很容易管理我们所需要的服务呢？真的很方便啦～ 他的功能其实很简单，只是直接在 /etc/rc.d/rc[0-6].d 里面针对某服务进行连结档案的设定而已。 例如上面的范例三，基本上，他仅是在 /etc/rc.d/rc3.d/, /etc/rc.d/rc4.d/ 及 /etc/rc.d/rc5.d/ 里面，建立一个连结档案， 该连结档案连结到 /etc/init.d/atd 里面就是了！这样说，可以理解吗？

既然这个玩意儿这么好用，那么我们可否将自己建立的服务给他加入 chkconfig 的管理当中？ 当然可以

啊！只是该服务必须要加入 init 可以管理的 script 当中，亦即是 /etc/init.d/ 当中才行。举个例子，我们在 /etc/init.d/ 里面建立一个 myvbird 档案， 该档案仅是一个简单的服务范例，基本上，没有任何用途.... 对于该档案的必须性是这样的：

- myvbird 将在 run level 3 及 5 启动；
- myvbird 在 /etc/rc.d/rc[35].d 当中启动时，以 S80 开始以 K70 结束。

那么我可以这样做：

```
[root@linux ~]# vi /etc/init.d/myvbird
#!/bin/bash
# chkconfig: 35 80 70
# description: 没啥！只是用来作为练习之用的一个范例
echo "Nothing"
# 这个档案很好玩喔！你可以参考你自己系统上面的档案；
# 基本上，比较重要的是第二行，他的语法是：
# chkconfig: [runlevels] [start number] [stop number]
# 其中， runlevels 为不同的 run level 状态， start number 与
# stop number 则是在 /etc/rc.d/rc[35].d 内建立以 S80myvbird
# 及 K70myvbird 为档名的设定方式！

[root@linux ~]# chkconfig --add myvbird
[root@linux ~]# chkconfig --list myvbird
myvbird         0:off   1:off   2:off   3:on    4:off   5:on    6:off
# 看吧！加入了 chkconfig 的管理当中了！再去看看 /etc/rc.d/ 底下的档案：

[root@linux ~]# find /etc/rc.d/ -type l | grep 'myvbird'|sort
/etc/rc.d/rc0.d/K70myvbird
/etc/rc.d/rc1.d/K70myvbird
/etc/rc.d/rc2.d/K70myvbird
/etc/rc.d/rc3.d/S80myvbird
/etc/rc.d/rc4.d/K70myvbird
/etc/rc.d/rc5.d/S80myvbird
/etc/rc.d/rc6.d/K70myvbird
# 很有趣吧！如果要将这些数据都删除的话，那么就下达这样的情况：
[root@linux ~]# chkconfig --del myvbird
[root@linux ~]# rm /etc/init.d/myvbird
```

chkconfig 真的是个不错用的工具吧！尤其是当你想要自己建立自己的服务时！ ^_^

- ntsysv

基本上， chkconfig 真的已经很好用了，不过，我们的 Fedora 还有提供一个更不错用的， 那就是 ntsysv 了！注意喔， chkconfig 很多的 distributions 都存在，但是 ntsysv 则是 Red Hat 系统特有的！

```
[root@linux ~]# ntsysv [--level <levels>]
```

```
参数：
--level ：后面可以接不同的 run level ，例如 ntsysv --level 35
范例：

范例一：直接编辑目前 run level 底下的开机预设启动项目：
[root@linux ~]# ntsysv
```




```
# 此时，你可以使用底下的按键来进行选择：
# 上下键： 可以在中间的方框当中，在各个服务之间移动；
# 空格键： 可以用来选择你所需要的服务，前面的 [*] 会有 * 出现；
# tab 键： 可以在方框、OK、Cancel 之间移动；
# [F1]键： 可以显示该服务的说明。举例来说，移动到 myvbird 按下 F1 后
```




```
# 没错！不要怀疑，这个就是 /etc/init.d/myvbird 档案内的 description:
# 后面接的内容啊！了解了吗？ ^_^
```

挺不错用的吧！还可以知道该服务的意义呢！也就是说， 如果你想要知道某个 /etc/init.d/ 底下的服务启动的信息为何， 直接以 vi 开启该档案，去察看一下 description: 的内容即可知道啊！ ^_^

各个服务的简单说明

随着 Linux 上面软件支持性越来越多，加上自由软件蓬勃的发展， 我们可以在 Linux 上面用的 daemons 真的越来越多了。所以，想要写完所有的 daemons 几乎是不可能的，因此，鸟哥这里仅介绍几个很常见的 daemons 而已， 更多的信息呢，就得要麻烦您自己使用 ntsysv 或者是 vi /etc/init.d/* 里面的档案去瞧一瞧啰～ ^_^

Stand Alone Daemons	
服务名称	• 参数档 • 预设启动的 port number • 鸟哥的建议：是否需要启动？
	基本说明
anacron	• /etc/anacrontab • 不需要使用 port • 全天候启用的主机，不需要开启这个服务
	当你的 Linux 主机并不是全天候开机的时候，这个 anacron 就可以帮你执行在『 crontab 』既定的时间内没有执行的工作！举个例子来说，当你的主机在晚上 12:00 会自动关闭，但是偏偏 crontab 这个例行性工作是在 4:00 工作， 这个时候例行性工作不是都没有做到吗？嗯！ anacron 就可以使用啦！

apmd	• /etc/sysconfig/apmd • 不需要使用 port • 一般主机不太需要这个 daemon
	apmd 是 Adventage Power Management daemon 的缩写，顾名思义，可以用来了解系统的『电池电量』， 如果对于手提式计算机才有需要吧我想!
atd	• /etc/at.allow, /etc/at.deny • 不需要使用 port • 若有启动 cron ，或许可以忽略
	这个总该不陌生了吧! 就是 仅进行一次的工作排程啰! 如果忘记了! 赶紧去查看一下!
autofs	• /etc/sysconfig/autofs • 不需要使用 port • 如果是服务器，不需要启动，如果是 Desktop，建议使用
	如果你的 Linux 是用来做为服务器的，那么这个服务就不需要启动了。 因为这个服务可以自动挂载很多的档案系统与装置，举例来说，自动挂载光盘啊、USB 硬盘啊等等的。 如果是主机，我们可以自己好好的控制，不需要系统自动挂载。如果是个人桌上型计算机， 那么启动这个 daemon 也不错!
crond	• /etc/crontab • 不需要使用 port • 务必启动啊!
	用来执行例行性命令的 daemon ，请务必要启动他!
cups	• /etc/printcap, /etc/cups/* • 预设使用 port 631 • 没有打印机的话，就不要启动
	这个服务在管理 Linux 主机上面的打印机的! 他可以用来作为本机打印机的管理，也可以用来管理网络打印机， 全名为 Common UNIX Printing System (CUPS)。如果您的网络环境当中有打印机， 而且想要透过 Linux 来提供给所有用户使用，那么就可以管理一下 cups 啰～
gpm	• /etc/sysconfig/mouse • 不需要使用 port • 不需要启动的
	在文字模式里面可以使用 mouse 来从事『复制、贴上、移动光标』等等的功能! 如果你是个教师，需要使用鼠标在纯文字接口底下秀出结果的话， 再使用这玩意就好了。基本上，不需要启动他!
httpd	• /etc/httpd/conf/httpd.conf, /etc/sysconfig/httpd • 使用 port 80 (and/or) 443 • 除非需要设定 WWW 服务器，否则不要启动
	这个玩意儿可有趣的很哩～一般来说，新手最喜欢架设 Web 网站啰， 而 WWW 服务器，就是这个玩意儿啊～更详细的信息请参考服务器篇的内容

iptables	• /etc/sysconfig/iptables • 不需要使用 port • 连上 Internet 的主机务必启动
	这个家伙就是 Linux 上面有名的『防火墙』啦～如果你的 Linux 是在区网内，而且没有连上 Internet 的话，那么这个防火墙机制可以暂时不要启动，因为可能会抵挡掉你主机所提供的服务。如果连上了 Internet ，不开这个，会死的很惨～
kudzu	• /etc/sysconfig/kudzu • 不需要使用 port • 如果系统已经稳定，不需要启动
	这个 daemon 预设是启动的，他会在开机的时候去侦测你的硬件，如果发现硬件有异动，或者是有新增其它的硬件，那么 kudzu 服务会主动的以 Fedora 相关的设定软件来设定你的新硬件。不过，对于稳定的系统来说，实在没有必要在开机的时候侦测一次硬件，因为....很慢～
named	• /etc/named.conf • 使用 port 53 • 不需要启动，除非是 DNS Server
	这是个很复杂的玩意儿，那就是 DNS (Domain Name System)。除非你真的很了解 DNS，否则这个服务不需要启动的！
netfs	• /etc/fstab • 不需要使用 port • 如果你的主机有预设挂载网络上的磁盘档案系统时，才开启。
	这个服务在自动的挂载 /etc/fstab 里头记录的关于网络档案系统，如 NFS, SMB (网芳) 等等，如果你的主机本身并没有挂载来自网络上的 filesystem，不需要启动。
network	• /etc/sysconfig/network, /etc/sysconfig/network-scripts/* • 不需要使用 port • 务必启动
	看檔名就知道啦！是用来管理网络的，所以，当然要启动了。不论你有没有网络卡，这个服务都要启动，因为至少 network 会驱动 lo 这个网络接口。更多的网络相关信息，参考服务器篇的内容。
nfs nfslock	• /etc/sysconfig/nfs • 随机使用 port，与 portmap 服务有关 • 不需要启动
	NFS 为 Network File System 的缩写，我们会在服务器篇谈这个服务，一般来说，不需要启动这个玩意儿～
ntpd	• /etc/ntp.conf, /etc/sysconfig/ntpd • 使用 port 123 • 不需要启动
	这个服务的全名是：Network Time Protocol，意思就是在进行网络校时的一个服

	务。 一般来说，不需要启动他。
portmap	• /sbin/portmap 直接启动 • 使用 port 111 • 除非你有启用类似 NFS 的服务，否则不需要启动
	这个咚咚与很多 RPC 的服务有关，例如 NFS 等等。一般来说，如果你的 Linux 尚未连上 internet， 这个服务不需要启动。不过，为了方便起见，各主要 linux distributions 都预设启动这个服务的。 鸟哥通常是关掉他！哈哈！
postfix sendmail	• /etc/postfix/* 或 /etc/mail/* • 使用 port 25 • 预设要启动
	这个就是邮件服务器啦！由于近来网络垃圾大增，所以，目前所有的 Linux distributions 预设都要启动 postfix 之类的邮件服务器后，我们发出的信件才会接收或者是传送。 预设的情况下，一定会有一个邮件服务器启动的，不要关闭他～ 否则我们主机上面的账号彼此之间无法以 email 传送数据喔！
smb	• /etc/samba/* • 使用 port 137~139, 445 等 • 不需要启动
	这个服务其实就是仿真 Linux 成为 Windows 的网络上的芳邻上头的主机啦～ 由于我们还没有连上 Internet 啊，所以自然不需要启动他啦～
sshd	• /etc/ssh/* • 使用 port 22 • 务必启动
	这个是取代 telnet 的远程联机服务器 daemon ，几乎所有的 Linux distributions 预设都会启动他～ 我们也可以透过这个玩意儿让远程主机联机进来啊！所以当然是启动的啊！
syslog	• /etc/syslog.conf • 不需要使用 port • 务必启动
	这个是登录文件记录的一个重要的 daemon ，没有他，你的主机几乎没有事后监控的功能～ 所以请务必启动。我们会在 认识登录档 当中来谈这个咚咚～
xfs	• 使用 /usr/sbin/chkfontpath 直接启动与侦测 • 使用 socket 7100 • 如果是纯文字接口，则不需要启动
	X Font Server, xfs，顾名思义，他是用来管理 X Window 的字形的一个服务， 如果你是 run level 5 ，或者是想要启动 X Window 的话，那么这个玩意儿就不能不启动。 不过，如果你跟鸟哥一样都是使用纯文字接口的话，这个玩意儿不用启动啦！
xinetd	• /etc/xinetd.d/*, /etc/xinetd.conf • 不一定，要看设定值

	• 务必启动
	刚刚上头才讲过这个玩意儿，不会就忘了吧？务必启动喔！
Super Daemons	
chargen chargen-udp	• 预设使用 port 19 • 不需要启动
	主要的功能在于提供类似远程打字的咚咚吧！
daytime daytime-udp	• 预设使用 port 13 • 不需要启动
	用来作为 daytime 的服务，这是 NTP (Network Time Protocol) 的上一代， 目的在进行时间的校正工作。不过，因为他不会计算网络联机过程当中的迟滞时间，并且是以明码传送，因此除了特殊目的外，目前已经很少使用这玩意儿了
krb5-telnet	• 预设使用 port 23 • 不需要启动
	用来取代传统的 telnet 服务！可提供 krb 5 的验证机制。
gssftp	• 预设使用 port 21 • 不必启动
	用来取代传统的 ftp server，可提供 krb 5 的验证机制

本章习题练习

（ 要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看 ）

- 使用 netstat -tul 与 netstat -tunl 有什么差异？为何会这样？

  使用 n 时， netstat 就不会使用主机名称与服务名称 (hostname & service_name) 来显示， 取而代之的则是以 IP 及 port number 来显示的。IP 的分析与 /etc/hosts 及 /etc/resolv.conf 有关， 这个在未来服务器篇才会提到。至于 port number 则与 /etc/services 有关，请自行参考喔！ ^_^

- 我想让系统一开机就自动执行 /usr/local/sbin/backup.sh 这个程序(假设已经存在)，你觉得可以如何进行？

  最简单的方法，就是直接将 /usr/local/sbin/backup.sh 这整个指令写入 /etc/rc.d/rc.local 档案当中即可！

认识与分析登录档

最近更新日期：2006/07/23

『登录档』似乎是常常听到的名词，网络上的老手们也常常告知新手们要多察看登录档， 那么这些登录档是干嘛用的？嗯！似乎是当你启发一个事件的时候，或者是有人登入你的 Linux 主机的时候，主机会有一些认证的程序或者是一些重要的讯息，由于这些讯息有被追踪的重要性，所以自然就有需要将他保留下来，以备未来的不时之需啰，这些讯息会被纪录在某些档案上， 这些档案就被称为登录档了！那么您晓得该登入者的信息被纪录在哪里吗？这些信息的量有多大呢？ 您可以每天自行观看吗？哇！如果能用 Shell Scripts 来分析的话，不是就更快速了吗？呵呵！ 这里鸟哥写了一个小小的分析档案 (logfile.sh)，让大家可以更快乐的管理你的 Linux 主机呦！ ^_^

什么是登录档？

这部分是最容易被新手所忽略的，那就是『 详细而确实的纪录或者是备份系统的登录文件 』。那么什么是登录档呢？简单的说，就是记录系统活动记录的几个档案， 例如：何时、何地 (来源 IP)、何人(login name)、做了什么动作， 另外就是系统在什么时候做了什么样的行为时，发生了什么样的事件等等，要知道的是，我们的 Linux 主机在背景之下，有相当多的 daemons 在工作着，那么这些工作中的程序总是会有一些讯息显示， 这些显示的讯息就是给记录在登录文件当中啦，也就是说，记录这些系统的重要讯息，就是登录文件所进行的纪录工作的内容了。

而由于这些记录的工作内容对于系统的信息太详细了， 若被取得将可能影响到系统的安全性，因此，通常这些登录档只有 root 可以进行视察的功能！ 那么为何要记录与解析登录文件呢？这是由于登录文件有几个重要的功能：

- 解决系统的错误：
  这个对于系统管理员来说是很重要的信息，例如：开机的过程当中侦测到的硬件讯息数据会记录到内存当中， 由于这些侦测的信息可以提供我们了解硬件信息，所以如果你的系统发生问题时，可以下达 dmesg 看看硬件的侦测有没有发生错误呢！另外，如果系统资源被耗尽、 核心活动发

生错误等等事件发生的时候，则系统登录文件亦会将错误的讯息记录在登录文件中 (通常是 /var/log/messages)，这些都可以藉以取得错误发生时的信息，并加以克服问题！！

- 解决网络服务的问题：
  在安装或设定新服务的套件时，最常使用到这个功能了！例如在安装启动 sendmail 时，如果 sendmail 无法提供服务的时候，那么无法提供服务的问题则会被纪录到登录文件当中去， 则只要分析登录档就可以了解问题点，并藉以解决问题啦！(所以我们常说『天助自助者』是真的啦！察看(1)屏幕上面的错误讯息与(2)登录文件的错误信息，几乎可以解决大部分的 Linux 问题！)
- 记录登录信息：
  这个东西相当的重要！例如：有天您的 apache 这个 WWW 服务挂了，你怎么知道何时挂掉的？ 而最后登入者是谁？！这都可以藉由分析 apache 的登录文件来取得信息； 此外，万一有一天您的系统被入侵，并且被利用来攻击他人的主机，这个时候对方的主机查出是您的 Linux 在进行攻击的行为，这个时候你要如何告知对方您的主机是由于被入侵所导致的问题， 并且协助对方继续往来源追查呢？！呵呵！此时登录档可是相当重要的呢！

因此，一个有经验的主机管理员，会随时随地查阅一下自己的登录文件，以随时掌握系统的最新脉动！ 那么常见的几个登录档有哪些呢？一般而言，有下面几个：

- /var/log/secure：
  记录登入系统存取数据的档案，例如 pop3, ssh, telnet, ftp 等都会记录在此档案中；
- /var/log/wtmp：
  记录登入者的讯息数据，由于本档案已经被编码过，所以必须使用 last 这个指令来取出档案的内容；
- /var/log/messages：
  这个档案相当的重要，几乎系统发生的错误讯息（或者是重要的信息）都会记录在这个档案中；
- /var/log/boot.log：
  记录开机或者是一些服务启动的时候，所显示的启动或关闭讯息；
- /var/log/maillog 或 /var/log/mail/*：
  纪录邮件存取或往来(sendmail 与 pop3)的使用者记录；
- /var/log/cron：
  这个是用来记录 crontab 这个例行性服务的内容的！
- /var/log/httpd, /var/log/news, /var/log/mysqld.log, /var/log/samba, /var/log/procmail.log：
  分别是几个不同的网络服务的记录文件啦！

常见的登录档就是这几个，但是，不同的 Linux distributions ，通常登录档的档名不会相同 (除了 /var/log/messages 之外)，所以说，您还是得要查阅您 Linux 主机上面的登录文件设定数据， 才能知道你的登录档主要档名喔！

好了，那么记录了这些登录文件之后，我要做什么分析呀！？基本上，一个好的系统管理员大概都知道 『一部主机负责的服务最好能少尽量少』，这是什么意思呢？ 也就是说，这部主机为邮件主机那么就专门负责邮件工作，不要还搞 WWW 服务！这样有几个好处，除了系统的安全性较佳之外 (因为开的 port 变少了！)， 登录档的解析也会比较简单！因为我们的 /var/log/secure 记录的登入者信息就会比较有一致性！ 那么我们就可以查询一下每日登入的使用者账号啦与错误讯息啦等等的！(当然啰，如果你的频宽够、经验丰富的话，那么一部主机上面安装所有的网络服务也是可以的啦！) 基本上，检查 /var/log/messages、/var/log/secure 这些个档案也就相当够了！ 因为系统发生的错误或者是警告讯息通常都会写入这些档案

中。

但是，如果我手边有数十部主机怎么办？我要不要一部一部去察看 log file 呢？呵呵！那样察看会死人ㄋㄟ～因此，我们底下也使用一个简易的登录档来分析 Red Hat (含 Fedora 啦！) 系列的登录文件吧！

在 Linux 的登录文件系统当中，大多以一支特定的 daemon 来进行写入这些讯息的工作，那就是 syslogd 这支程序啦！所以说，只要软件套件有支持 syslogd 的登录文件写入模式， 那么该软件套件的信息就会被写入到 syslogd 管理的登录档当中。

另外，由于登录档如果一直长大的话，那么这些登录档的写入动作将会很没有效率， 这是因为档案太大时，ASCII 格式码的数据文件写入比较麻烦的缘故！ 那么怎么进行登录文件数据的备份工作呢？呵呵！那就使用 logrotate 吧！将数据进行轮转(rotate)？什么是轮转？！(我ㄌㄟ台语不轮转呦！？) 其实也可以称为轮替啦！

所谓的 logrotate 基本上，就是将旧的 log 档案更改名称，然后建立一个空的 log 档案，如此一来，新的 log 档案将从零开始记录，然后只要将旧的 log 档案留下一阵子，嗯！那就可以达到将登录档『轮转』的目的啦！ 此外，如果旧的纪录 (大概要保存几个月吧！保存了一段时间没有问题，那么就可以让系统自动的将他砍掉， 免得占掉很多宝贵的硬盘空间说！(举个例子来说，鸟哥的 WWW 网站一个月的登录档，所占掉的硬盘空间大小，大概就有 3GB 这么多....而且都是纯文字文件....很可怕吧！)

所以说，基本上，针对 log 档案来设计的服务有这两支：

- syslogd：
  进行系统或者是网络服务的登录文件记录工作；
- logrotate：
  将旧的数据更名，并且建立新的登录档， 以保持登录档的『新鲜』，并视设定将最旧的登录档删除。

所以，接着下来我们来谈一谈怎么样规划这两支程序呢？！就由 syslogd 这支程序先谈起吧！毕竟得先有登录档，才可以进行 logrotate 呀！您说是吧！？

登录文件的纪录： syslogd

刚刚上面提到说，Linux 的登录档主要是由 syslogd 这个 daemon 在负责，那么您的 Linux 是否有启动 syslogd 呢？而且是否有设定开机时启动呢？呵呵！检查一下先：

```
[root@linux ~]# ps aux | grep syslog
USER     PID %CPU %MEM   VSZ   RSS TTY  STAT START  TIME COMMAND
root   11129  0.0  0.0  1616   204 ?    Ss   Oct03  0:01 syslogd -m 0
# 瞧！对吧！确实有启动的！

[root@linux ~]# chkconfig --list | grep syslog
syslog   0:off   1:off   2:on   3:on   4:on   5:on   6:off
# 因为鸟哥的系统用 run level 3 ，所以，也是有启动的！
```

看到 syslog 这个服务名称了吧？！呵呵！所以知道他已经在背景底下工作啰！如前所述， syslog 这支程序提供了『系统登入信息记录』及『 Kernel 错误或警示信息记录』等功能，此外， 他还提供了『本地端与远程计算机的登录信息记录』功能，所以，可以将远程的主机登入信息同时记录在本地端呢！ 很不错的功能吧！！此外，目前正规使用的系统服务中，大都预设支持以 syslog 这一个服务来记录他的登录档案数据，例如 apache, samba, sendmail 等等。

登录档内容的一般格式

一般来说，通常经过 syslog 而记录下来的数据主要有：

- 事件发生的日期与时间；
- 发生此事件的主机名称；
- 启动此事件的服务名称（如 samba, xinetd 等）或函式名称（如 libpam ..）；
- 该讯息数据内容。

当然，这些信息的『详细度』是可以修改的，而且，这些信息可以作为系统除错之用呢！ 我们先来看一下 /var/log/secure 的内容显示些什么呢？

```
[root@linux ~]# cat /var/log/secure
Oct 16 10:16:13 linux sshd[3494]: Accepted password for dmtsai from
192.168.1.11 port 1037 ssh2
Oct 16 10:20:15 linux xinetd[21592]: START: shell pid=4176 from=192.168.1.31
Oct 16 14:50:25 linux sshd[26665]: Accepted password for dmtsai from
192.168.1.11 port 1078 ssh2
Oct 16 19:56:06 linux xinetd[20576]: START: shell pid=21013 from=192.168.1.31
```

以第一笔数据为例，该数据说明的内容为：『时间在 Oct 16 10:16:13 (10/16, 10:16)时；由主机名称为 linux 的那部主机当中；由 sshd (且其 PID 为 3494) 那项服务所产生的一个讯息；讯息内容说明是：接受来自 192.168.1.11 连接至本机，使用 ssh2 联机机制，接受的使用者为 dmtsai』。有够清楚的吧！ ^_^。就是因为太清楚了， 包括那个 192.168.1.11 的 IP 来源，以及使用者账号为 dmtsai ， 这些信息如果让比较高竿的 cracker 知道后，是有可能猜测密码的， 所以啰，这些信息当然不能够外流啦！好好保存吧！

其实还有很多的信息值得查阅的呢！尤其是 /var/log/messages 的内容。记得一个好的系统管理员， 要常常去『巡视』登录档的内容喔！尤其是：

- 当你觉得系统似乎不太正常时；
- 某个 daemon 老是无法正常启动时；
- 某个使用者老是无法登入时；
- 某个 demon 执行过程老是不顺畅时；

还有很多啦！反正觉得系统不太正常，就得要查询查询登录档就是了。

登录档的设定档：/etc/syslog.conf

什么？登录档还有设定档？？喔!不是啦～是 syslogd 这个 daemon 的设定档啦！我们现在知道 syslogd 可以负责主机产生的各个信息的登录，而这些信息本身是有『严重等级』之分的， 而且，这些数据要传送到什么文件名的档案里去？这都是可以修订的呢，所以我们才会在一开头的地方讲说， 每个 Linux distributions 放置的登录档档名可能会有所差异啊！

基本上， syslog 针对各种服务与讯息记录在某些档案的设定档就是：

/etc/syslog.conf

这个档案规定了『哪些服务需要被纪录， 该服务产生的什么等级的讯息要被纪录？』这个 /etc/syslog.conf 的内容语法是这样的：

```
服务名称[.=!]讯息等级                    讯息记录的文件名或装置或主机
# 例如底下：
mail.info                       /var/log/maillog_info
```

简单的说明如下：

- 服务名称：

什么服务产生的讯息要被纪录的意思。syslog 认识的服务主要有底下这些：

- auth, authpriv：主要与认证有关的机制，例如 telnet, login, ssh 等需要认证的服务都是使用此一机制；
- cron：就是例行性命令 cron/at 等产生讯息记录的地方；
- daemon：与各个 daemon 有关的讯息；
- kern：就是核心 (kernel) 产生讯息的地方；
- lpr：亦即是打印相关的讯息啊！
- mail：只要与邮件收发有关的讯息纪录都属于这个；
- news：与新闻群组服务器有关的东西；
- syslog：就是 syslogd 这支程序本身产生的信息啊！
- user, uucp, local0 ~ local7：与 Unix like 机器本身有关的一些讯息。

基本上，syslog 所认识的信息服务与一般我们常说的服务不太一样。举例来说， 关于邮件服务器，我们可以选择 sendmail, qmail 或者是当红的 postfix 这些软件来达成， 但这些服务器使用的都是同一个通讯协议，亦即是 smtp 这个玩意儿 (参阅一下 /etc/services 的内容， 找到 smtp 观察一下 port number 啦！)。所以，这些同性质的服务器，产生的讯息都属于 syslog 当中的 mail 所管辖的范围喔！

另外，每种服务所产生的数据量其实是差异很大的，举例来说， mail 的登录文件讯息多的要命， 每一封信件进入后， mail 至少需要记录『寄信人的信息；与收信者的讯息』等等， 而如果是用来做为工作站主机的，那么登入者 (利用 login 登录主机处理事情) 的数量一定不少，那个 authpriv 所管辖的内容可就多的要命了。

为了让不同的信息放置到不同的档案当中，好让我们分门别类的进行登录档的管理， 所以啰，将各种类别的服务之登录文件，记录在不同的档案里面，就是我们 /etc/syslog.conf 所要作的规范了！

- 讯息等级

每种服务所产生的讯息是有差异的，有启动时告知系统的信息讯息 (information)， 有被入侵时发出的警告讯息 (warn) ，还有系统硬件发生错误时，所产生的重大问题讯息 (error 等等)； 基本上，系统将讯息分为七个主要的等级，依序是这样的(由不重要排列到重要讯息等级)：

1. info：仅是一些基本的讯息说明而已；
2. notice：比 info 还需要被注意到的一些信息内容；
3. warning 或 warn：警示的讯息，可能有问题， 但是还不至于影响到某个 daemon 运作的信息；基本上， info, notice, warn 这三个讯息都是在告知一些基本信息而已，应该还不至于造成一些系统运作困扰；
4. err 或 error ：一些重大的错误讯息， 例如设定文件的某些设定值造成该服务服法启动的信息说明， 通常藉由 err 的错误告知，应该可以了解到该服务无法启动的问题呢！
5. crit：比 error 还要严重的错误信息，这个 crit 是临界点 (critical) 的缩写，这个错误已经很严重了喔！
6. alert：警告警告，已经很有问题的等级，比 crit 还要严重！
7. emerg 或 panic：疼痛等级，意指系统已经几乎要当机的状态！ 很严重的错误信息了。 通常大概只有硬件出问题，导致整个核心无法顺利运作，就会出现这样的等级的讯息吧！

除了这些有等级的讯息外，还有两个特殊的等级，那就是 debug(错误侦测等级) 与 none(不需登录等级) 两个，当我们想要作一些错误侦测，或者是忽略掉某些服务的信息时，就用这两个咚咚吧！

特别留意一下在讯息等级之前还有 [.=!] 的连结符号喔！他代表的意思是这样的：

- . ：代表『比后面还要高的等级(含该等级)都被记录下来』的意思， 例如： mail.info 代表只要是 mail 的信息，而且该信息等级高于 info (含 info 本身)时，就会被记录下来的意思。
- .=：代表所需要的等级就是后面接的等级而已，其它的不要！
- .!：代表不等于，亦即是除了该等级外的其它等级都记录。

一般来说，我们比较常使用的是『.』这个连结符号啦！^_^

- 讯息记录的文件名或装置或主机

再来则是这个讯息要放置在哪里的纪录了。通常我们使用的都是记录的档案啦！但是也可以输出到装置呦！例如打印机之类的！也可以记录到不同的主机上头去呢！底下就是一些常见的放置处：

- 档案的绝对路径：通常就是放在 /var/log 里头的档案啦！
- 打印机或其它：例如 /dev/lp0 这个打印机装置
- 使用者名称：显示给使用者啰！
- 远程主机：例如 @test.adsldns.org 当然啦，要对方主机也能支持才行！
- *：代表『目前在在线的所有人』，类似 wall 这个指令的意义！

基本上，整个 syslog 的设定档就只是这样而已，底下我们来思考一些例题， 好让你可以更清楚的知道如何设定 syslogd 啊！

例题：如果我要将我的 mail 相关的数据给他写入 /var/log/maillog 当中，那么在 /etc/syslog.conf 应该如何写？ 答：

基本的写法是这样的：

```
mail.info         /var/log/maillog
```

注意到上面喔，当我们的等级使用 info 时，那么『任何大于 info 等级(含 info 这个等级)之上的讯息，都会被写入到后面接的档案之中！』这样可以了解吗？也就是说，我们可以将所有 mail 的登录信息都纪录在 /var/log/maillog 里面的意思啦！

例题：我要将新闻群组数据 (news) 及例行性命令的信息 (cron) 都写入到一个称为 /var/log/cronnews 的档案中，但是这两个程序的警告讯息记录在 /var/log/cronnews.warn 该如何设定我的档案呢？

答：

很简单啦！既然是两个程序，那么只好以分号来隔开了，此外，由于第二个指定档案中，我只要记录警告讯息，因此设定上需要指定『=』这个符号，所以就成为了：

```
news.*;cron.*          /var/log/cronnews
news.=warn;cron.=warn  /var/log/cronnews.warn
```

上面那个『.=』就是在指定等级的意思啦！由于指定了等级，因此，只有这个等级的讯息才会被纪录在这个档案里面呢！

例题：我的 messages 这个档案需要记录所有的信息，但是就是不想要记录 cron, mail 及 news 的信息，那么应该怎么写才好？

答：

可以有两种写法，分别是：

```
*.*;news,cron,mail.none            /var/log/messages
*.*;news.none;cron.none;mail.none  /var/log/messages
```

使用『,』分隔时，那么等级只要接在最后一个即可，如果是以『;』来分的话，那么就需要将服务与等级都写上去啰！这样会设定了吧！

了解语法之后，我们来看一看在尚未开启网络服务的情况下，我们的 syslog 有哪些系统服务已经在纪录了呢？！那就是瞧一瞧 /etc/syslog.conf 这个档案的预设内容啰！（ 注意！如果需要将该行做为批注时，那么就加上 # 符号就可以啦！）

```
# 来自 Fedora Core Release 4 的相关资料
[root@linux ~]# vi /etc/syslog.conf
#kern.*                                    /dev/console
# 只要是 kernel 产生的讯息，全部都送到 console 去！
# 这个项目预设是关闭的！不过，只要您愿意，可以开启就是了！
```

```
*.info;mail.none;authpriv.none;cron.none    /var/log/messages
# 在已知各服务的讯息中，不要记录到这个档案当中啦！
# 例如 mail 我们已经预设要记录在 /var/log/maillog （底下），
# 所以自然不要在记录 mail 到这个 /var/log/messages 档案中啦！
# 然后其它的讯息全部都记录到 /var/log/messages 当中！
# 所以这个档案相当的重要！没有被规定到的讯息都可以在这里找到！

authpriv.*                                  /var/log/secure
# 这个就是经过一些身份确认的行为之后，需要记录身份的档案啦！
# 什么是身份确认呢？例如 pop3 收信要输入账号与密码、 ssh 与 telnet, ftp
# 等等的服务需要输入账号与密码，这些都会在 /var/log/secure 里面记录！
# 他可也是相当重要的一个档案呢！

mail.*                                      -/var/log/maillog
# 只要跟 mail 有关的（不论是 pop3 还是 sendmail ）都会被纪录到这个档案中！

cron.*                                      /var/log/cron
# 还记得例行性命令那一章节吗？！对啦！就是那个 crontab 的东西，
# 那东西的服务程序名称就是 cron 啦！执行 cron 的结果都记录于此！

*.emerg                                     *
# 任何时候发生的警告讯息都会显示给在线的所有人！那个 * 就是目前在线的所有人啦！
# 这个就是利用 wall 之类的方式传输讯息的啊！

uucp,news.crit                              /var/log/spooler
# 记录新闻错误高于 crit 的等级的信息，写入 spooler 当中！

local7.*                                    /var/log/boot.log
# 将开机的当中的讯息给他写入 /var/log/boot.log 这个档案当中呦！
```

基本上，每个版本的 syslog.conf 差异是颇大的，所以，每个登录文件记录的数据其实不很固定。 举例来说，如果您曾经使用过 Mandriva 的话，他的 syslogd 设定信息有点像这样：

```
# 来自 Mandrake 9.0 的 syslog.conf 内容
[root@linux ~]# vi /etc/syslog.conf
auth,authpriv.*                               /var/log/auth.log
*.*;auth,authpriv.none                       -/var/log/syslog
user.*                                       -/var/log/user.log
*.info;mail.none;;news.none;authpriv.none    -/var/log/messages
Authpriv.*                                    /var/log/secure
mail.=debug;mail.=info;mail.=notice          -/var/log/mail/info
mail.=warn                                   -/var/log/mail/warnings
mail.err                                     -/var/log/mail/errors
cron.=debug;cron.=info;cron.=notice          -/var/log/cron/info
cron.=warn                                   -/var/log/cron/warnings
```

```
cron.err                                          -/var/log/cron/errors
kern.=debug;kern.=info;kern.=notice               -/var/log/kernel/info
kern.=warn                                        -/var/log/kernel/warnings
kern.err                                           /var/log/kernel/errors
lpr.=debug;lpr.=info;lpr.=notice                  -/var/log/lpr/info
lpr.=warn                                         -/var/log/lpr/warnings
lpr.err                                           -/var/log/lpr/errors
news.=debug;news.=info;news.=notice               -/var/log/news/news.notice
news.=crit                                        -/var/log/news/news.crit
news.=err                                         -/var/log/news/news.err
Daemon.=debug;daemon.=info;daemon.=notice         -/var/log/daemons/info
Daemon.=debug;daemon.=info;daemon.=notice         -/var/log/daemons/info
Daemon.=warn                                      -/var/log/daemons/warnings
Daemon.err                                        -/var/log/daemons/errors
*.emerg                                           *
Local1.*                                          -/var/log/explanations
```

基本上,他将每个服务的登录档都分成三个内容,那么我们就可以简单的依据比较严重的登录档来解析，有助于系统的快速整理。但是每个人的喜好不同，因此，并不见得这样的设定大家都喜欢。 而像鸟哥自己，我自己写了一支分析 logfile 的程序，该程序主要针对 Red Hat 系统的登录文件来作处理的，则 Mandriva 的版本是否适用呢？当然适用啊！ 只要手动修订一下 /etc/syslog.conf 内容即可啊！这样说，瞭了吧？！ ^_^

另外,这些档案都相当的重要 (例如什么时候被谁登入进来主机啦!?),所以他们的权限大多是属于 root 的可擦写而已！这点非常需要小心而留意！ (请注意，在系统的预设状况中，所有的未知状态的讯息几乎都是写入 /var/log/messages 这个档案中，所以，如果系统有问题，请详细的检查一下这个 /var/log/messages 档案吧！！)

如果您有其它的需求，所以需要特殊的档案来帮你记录时，呵呵！别客气，千万给他记录在 /etc/syslog.conf 当中，如此一来，您就可以重复的将许多的信息记录在不同的档案当中，以方便您的管理呢！

让我们来作个练习题吧!如果你想要让『所有的信息』都额外写入到 /var/log/admin.log 这个档案时， 你可以怎么作呢？先自己想一想，并且作一下，再来看看底下的作法啦！

```
# 1. 先设定好所要建立的档案设置!
[root@linux ~]# vi /etc/syslog.conf
*.info          /var/log/admin.log

# 2. 重新启动 syslog 呢!
[root@linux ~]# /etc/init.d/syslog restart
[root@linux ~]# ll /var/log/admin.log
-rw-------  1 root root 122 Oct 23 22:21 /var/log/admin.log
# 瞧吧! 建立了这个档案出现啰!
```

很简单吧！如此一来，所有的信息都会写入 /var/log/admin.log 里面了！

登录档的安全性设置

好了，由上一个小节里面我们知道了 syslog.conf 的设定，也知道了登录档内容的重要性了， 所以，如果幻想你是一个很厉害的黑客，想利用他人的计算机干坏事，然后又不想留下证据， 你会怎么作？对啦！就是离开的时候将屁股擦干净，将所有可能的讯息都给他抹煞掉， 所以第一个动脑筋的地方就是登录档的清除工作啦～

哇！鸟哥教人家干坏事……喂！不要乱讲话～俺的意思是，如果改天你发现你的登录档不翼而飞了， 或者是发现你的登录档似乎不太对劲的时候，最常发现的就是网友常常会回报说，他的 /var/log 这个目录『不见了！』不要笑！这是真的事情?请记得，『赶快清查你的系统！』

伤脑筋呢！那么有没有办法防止这样的事情呢？有呀！拔掉网络线……呵呵！别担心，基本上， 我们可以透过一个隐藏的属性来设定你的登录档，成为『 只可以增加数据，但是不能被删除 』的状态，那么或许可以达到些许的保护！不过，如果你的 root 账号被破解了，那么底下的设定还是无法保护的，因为你要记得『 root 是可以在系统上面进行任何事情的 』，因此，请将你的 root 这个账号的密码设定的安全一些！千万不要轻忽这个问题呢！

好了，开始来设定一下基本的隐藏属性吧！那就是在 Linux 档案属性 提过的 lsattr 与 chattr 这两个东西啦！如果将一个档案以 chattr 设定 i 这个属性时，那么该档案连 root 都不能杀掉！而且也不能新增数据，嗯！真安全！但是，如此一来登录文件的功能岂不是也就消失了？ 因为没有办法写入呀！所以啰，我们要使用的是 a 这个属性！你的登录文件如果设定了这个属性的话，那么 他将只能被增加，而不能被删除！嗯！ 这个项目就非常的符合我们登录档的需求啦！因此，您可以这样的增加你的登录文件的隐藏属性。

> Tips:
> 请注意，底下的这个 chattr 的设定状态：『仅适合已经对 Linux 系统很有概念的朋友』来设定， 对于新手来说，建议您直接使用系统的默认值就好了，免得到最后登录档无法写入～ 那就比较糗一点！ @_@



```
[root@linux ~]# chattr +a /var/log/messages
[root@linux ~]# lsattr /var/log/messages
-----a------- /var/log/messages
```

加入了这个属性之后，你的 /var/log/messages 登录档从此就仅能被增加，而不能被删除，直到 root 以『 chattr -a /var/log/messages 』取消这个 a 的参数之后，才能被删除或移动喔！

虽然，为了您登录文件的信息安全，这个 chattr 的 +a 旗标可以帮助您维护好这个档案， 不过，如果您的系统已经被取得 root 的权限，而既然 root 可以下达 chattr -a 来取消这个旗标， 所以啰，还是有风险的啦！此外，前面也稍微提到，新手最好还是先不要增加这个旗标， 很容易由于自己的忘记，导致系统的重要讯息无法记录呢。

基本上，鸟哥认为，这个旗标最大的用处除了在保护您登录文件的数据外， 他还可以帮助您避免掉不小心写入登录档的状况喔。要注意的是，当『 你不小心 "手动" 更动过登录档后，例如那个

/var/log/messages ， 你不小心用 vi 开启他，离开却下达 :wq 的参数，呵呵！那么该档案未来将不会再继续进行登录动作！ 』这个问题真的很常发生！由于你以 vi 储存了登录档，则 syslogd 会误判为该档案已被更动过， 将导致 syslogd 不再写入该档案新的内容～很伤脑筋的！

要让该登录档可以继续写入，你只要重新启动 syslog (/etc/init.d/syslog restart) 即可。 不过，总是比较麻烦。所以啊，如果你针对登录档下达 chattr +a 的参数，嘿嘿！ 未来你就不需要害怕不小心更动到该档案了！因为无法写入嘛！除了可以新增之外～ ^_^

不过，也因为这个 +a 的属性让该档案无法被删除与修改，所以啰，当我们进行登录档案轮替时 (logrotate) ，将会无法移动该登录档的档名呢！所以会造成很大的困扰。这个困扰虽然可以使用 logrotate 的设定档来解决，但是，还是先将登录档的 +a 旗标拿掉吧！

```
[root@linux ~]# chattr -a /var/log/messages
```

登录文件主机的简单设定

我们在之前稍微提到的，在 syslog.conf 档案当中，可以将登录数据传送到打印机， 或者是远程主机上面去。这样做有什么意义呢？ 如果你将登录信息直接传送到打印机上面的话，那么万一不小心你的系统被 cracker 所入侵， 他也将您的 /var/log/ 砍掉了，怎么办？没关系啊！反正你已经将重要数据直接以打印机记录起来了， 嘿嘿！他是无法逃开的啦！^_^

再想象一个环境，你的办公室内有十部 Linux 主机，每一部负责一个网络服务， 你为了要了解每部主机的状态，因此，你常常需要登入这十部主机去查阅你的登录档～ 哇！光用想的，每天要进入十部主机去查数据，想到就烦～没关系～这个时候我们可以让某一部主机当成 『登录文件主机』 ，用他来记录所有的十部 linux 主机的信息，嘿嘿！这样我就直接进入一部主机就可以了！ 省时又省事，真方便～

那要怎么达到这样的功能呢？首先，你必须要连上网络嘛！不然怎么接受信息？ 再来，由于 syslog 用的是 udp 封包的 514 埠号,因此,你的登录文件主机就得要启用这个 port 才行～ 如何启用呢？很简单啊！这样做就可以了！

```
# 1. 先取得 port number 的信息！
[root@linux ~]# grep 514 /etc/services
syslog          514/udp
# 特别特别注意，你的 /etc/services 里面必须要存在这一行才行～
# 如果不存在这一行，你可以手动自行增加的！

# 2. 修改 syslogd 的启动设定档，通常在 /etc/sysconfig 内！
[root@linux ~]# vi /etc/sysconfig/syslog
# 找到底下这一行：
SYSLOGD_OPTIONS="-m 0"
# 改成底下这样子！
SYSLOGD_OPTIONS="-m 0 -r"

# 3. 重新启动与观察 syslogd 喔！
[root@linux ~]# /etc/init.d/syslog restart
```

```
[root@linux ~]# netstat -tlunp
Proto Recv-Q Send-Q Local Address Foreign Address State   PID/Program name
udp    0       0       0.0.0.0:514    0.0.0.0:*              24314/syslogd
# 嘿嘿！你的登录文件主机已经设定妥当啰！很简单吧！
```

透过这个简单的动作，你的 Linux 主机已经可以接收来自其它主机的登录信息了！ 当然啦，你必须要知道网络方面的相关基础，这里鸟哥只是先介绍，未来了解了网络相关信息后，再回头来这里瞧一瞧先！^_^，此外， 更多的相关信息可以 man syslogd 查阅看看喔！ ^_^

至于 client 端的设定就简单多了！只要指定某个信息传送到这部主机即可！ 举例来说，我们的登录文件主机 IP 为 192.168.1.100 ，而 client 端希望所有的数据都送给主机，所以，可以在 /etc/syslog.conf 里面新增这样的一行：

```
[root@linux ~]# vi /etc/syslog.conf
*.*        @192.168.1.100
```

立刻就搞定了！而未来主机上面的登录文件当中，每一行的『主机名称』就会显示来自不同主机的信息了。很简单吧！ ^_^。接下来，让我们来谈一谈，那么如何针对登录档来进行轮转 (rotate) 呢？

登录档的轮替(logrotate)：

好了！假设我们已经将登录数据写入了记录文件中了，也已经利用 chattr 设定了 +a 这个属性了，那么该如何进行 logrotate 的工作呢！？这里请特别留意的是：『syslog 利用的是 demand 的方式来启动的，当有需求的时候立刻就会被执行的，但是 logrotate 却是在规定的时间到了之后才来进行登录档的轮替，所以这个 logrotate 程序当然就是挂在 cron 底下进行的呦！』 仔细看一下 /etc/cron.daily/ 里面的档案，嘿嘿~看到了吧！ /etc/cron.daily/logrotate 就是记录了每天要进行的登录档轮替的行为啦！^_^！底下我们就来谈一谈怎么样设计这个 logrotate 吧！

---

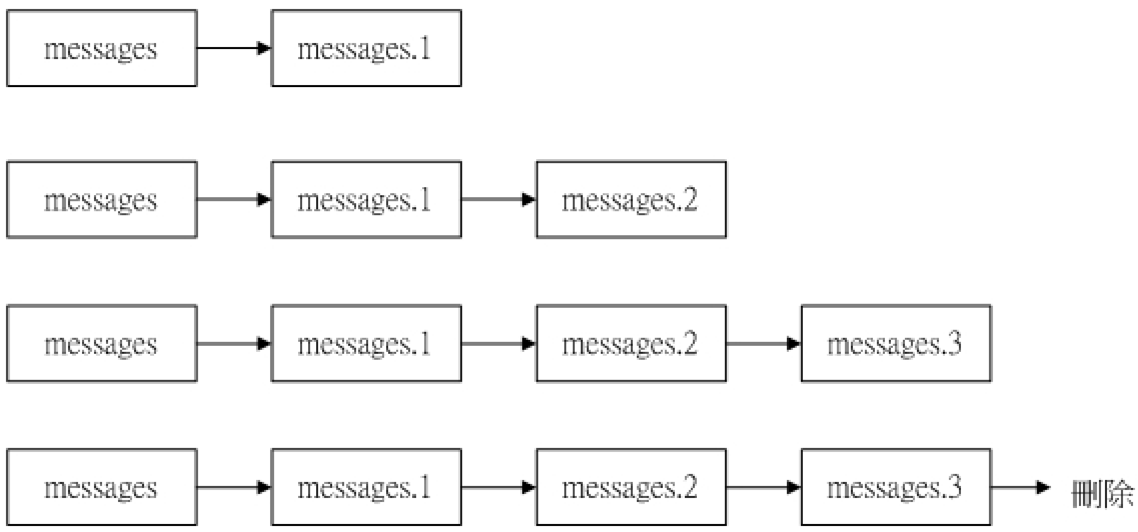
logrotate 的设定档

既然 logrotate 主要是针对登录档来进行轮替的动作，所以啰，他当然必须要记载『 在什么状态下才将登录档进行轮替』的设定啊！那么 logrotate 这个程序的参数设定文件在哪里呢？！呵呵！那就是：

- /etc/logrotate.conf
- /etc/logrotate.d/

注意啰！那个 logrotate.conf 才是主要的参数档案，至于 logrotate.d 是一个目录， 该目录里面的所有档案都会被主动的读入 /etc/logrotate.conf 当中来进行！另外，在 /etc/logrotate.d/ 里面的档案中，如果没有规定到的一些细部设定，则以 /etc/logrotate.conf 这个档案的规定来指定为默认值！

好了，刚刚我们提到 logrotate 的主要功能就是将旧的登录档案移动成旧档， 并且重新建立一个新的空的档案来记录，他的执行结果有点类似底下的图示：



图一、登录档进行 logrotate 的结果

由上面的图示我们可以清楚的知道，当第一次执行完 rotate 之后，原本的 messages 会变成 messages.1 而且会制造一个空的 messages 给系统来储存登录文件。而第二次执行之后，则 messages.1 会变成 messages.2 而 messages 会变成 messages.1 ，又造成一个空的 messages 来储存登录档！那么如果我们仅设定保留三个登录档而已的话，那么执行第四次时，则 messages.3 这个档案就会被删除，并由后面的较新的保存登录档所取代！基本的工作就是这样啦！

那么多久进行一次 logrotate 的工作呢？嗯！这些都记录在 logrotate.conf 里面，我们来看一下预设的 logrotate 的内容吧！

```
[root@linux ~]# vi /etc/logrotate.conf
# 底下的设定是 "logrotate 的预设设定值" ，如果个别的档案设定了其它的参数，
# 那么将以个别的档案设定为主，若该档案没有设定到的参数，
# 则以这个档案的内容为默认值！

weekly
# 预设每个礼拜对登录档进行一次 rotate 的工作

rotate 4
# 保留几个登录档呢？预设是保留四个！

create
# 是否建立新的登录文件来记录呢？因为我们要继续记录，所以当然是建立啰！

#compress
# rotate 之后的登录档，要不要压缩，通常是不要压缩啦，但是如果你的系统很忙碌，
# 表示你的登录档很庞大的时候，那么最好就是给他压缩一下比较不会占空间！
# 如果要压缩，将 # 拿掉即可！不过，登录档扩展名会变成 messages.1.gz 喔！
```

```
include /etc/logrotate.d
# 将底下这个目录中的所有档案都读进来执行 rotate 的工作！

/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}
# 基本上，在 logrotate.conf 档案当中，只有这个数据是在记载如何对登录文件进行轮替的！
# 这个登录文件记载的就是使用 login 登入系统时的使用者状态，还记得那个 last 指令吧？
# 就是读自 /var/log/wtmp 当中记录的数据啦！整个段落的意义是：
# 1. 每个月进行一次 log rotate 的工作；
# 2. 将档案的权限设定为 664 ，且拥有者为 root ，群组为 utmp；
# 3. 仅保存前一个月的 rotate 备份！
# 这也是为什么我们说 last 只会秀出这个月的数据而已的原因～因为一个月轮替一次嘛！
# 这个 rotate 可以改大一点，例如 5 以保存五个月，以利追踪

/var/log/btmp {
    missingok
    monthly
    create 0600 root utmp
    rotate 1
}
# 这个跟 wtmp 类似！
```

由这个档案的设定我们可以知道 /etc/logrotate.d 其实就是由 /etc/logrotate.conf 所规划出来的目录，所以，其实我们可以将所有的资料都给他写入 /etc/logrotate.conf 即可，但是这样一来这个档案就实在是太复杂了，尤其是当我们使用很多的服务在系统上面时， 每个服务都要去修改 /etc/logrotate.conf 的设定也似乎不太合理～ 所以，如果独立出来一个目录，那么每个以 RPM 打包方式所建立的服务的登录档轮替设定， 就可以独自成为一个档案，并且放置到 /etc/logrotate.d/ 当中即可， 真是方便又合理的做法啊！ ^_^

一般来说，这个 /etc/logrotate.conf 是『预设的轮替状态』而已， 我们的各个服务都可以拥有自己的登录档轮替设定，您也可以自行修改成自己喜欢的样式啊！ 例如，如果您的系统的空间够大，并且担心除错以及黑客的问题，那么可以：

- 将 rotate 4 改成 rotate 9 左右，以保存较多的备份文件；
- 大部分的登录档不需要 compress 啰！但是空间太小就需要 compress ！尤其是很占硬盘空间的 httpd 更需要 compress 的！

好了，上面我们大致介绍了 /var/log/wtmp 这个档案的设定，但是还是不很详细啦，所以底下我们以 /etc/logrotate.d/syslog 这个轮替 syslog 服务的档案，来看看该如何设定他的 rotate 呢：

```
[root@linux ~]# vi /etc/logrotate.d/syslog
/var/log/messages /var/log/secure /var/log/maillog /var/log/spooler
```

```
/var/log/boot.log /var/log/cron {
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
    endscript
}
# 亦即是这样的格式啦！
# 登录文件的绝对路径文件名 {
#          各项基本上设定值
# }
```

在上面的语法当中，我们知道正确的 logrotate 的写法为：

- 将要被处理的登录档档名（包含绝对路径）写在前面，可以使用空格符分隔多个登录档；
- 用 { } 包括所有的设定；
- 设定的项目与前面提到的相同，并且可加入轮替前 (pre) 与后 (post) 的一些特殊执行的指令！这个设定需与 sharedscripts .... endscript 设定合用才行；
    - prerotate：在启动 logrotate 之前进行的指令，例如修改登录文件的属性等动作；
    - postrotate：在做完 logrotate 之后启动的指令，例如重新启动 (kill -1 或 kill -HUP) 某个服务！
    - Prerotate 与 postrotate 对于已经加上了特殊属性的档案处理上面，是相当重要的执行程序！

也就是说，这一段设定值说明的是：『 /var/log 目录内的 messages, secure, mailog, spooler, boot.log 及 cron 这六个档案，每个礼拜进行一次轮替，且保留四个登录档，此外， 在轮替进行完毕之后，执行 syslog 的重新启动』为什么会知道每个礼拜进行一次呢？ 呵呵！因为没有提到该设定项目，所以就用 /etc/logrotate.conf 内的默认值来作用啊！

好！若假设我们有针对 /var/log/messages 这个档案增加 chattr +a 的属性存在时， 依据 logrotate 的工作原理，我们知道，这个 /var/log/messages 将会被更名成为 /var/log/messages.1 才是。但是由于加上这个 +a 的参数啊，所以，更名是不可能成功的！ 那怎么办呢？呵呵！就利用 prerotate 与 postrotate 来进行登录档轮替前、后所需要作的动作啊！ 果真如此时，那么你可以这样修改一下这个档案喔！

```
[root@linux ~]# vi /etc/logrotate.d/syslog
/var/log/messages /var/log/secure /var/log/maillog /var/log/spooler
/var/log/boot.log /var/log/cron {
    sharedscripts
    prerotate
          /usr/bin/chattr -a /var/log/messages
    endscript
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
          /usr/bin/chattr +a /var/log/messages
```

```
    endscript
}
```

看到否？就是先给他去掉 a 这个属性，让登录文件 /var/log/messages 可以进行轮替的动作， 然后执行了轮替之后，再给他加入这个属性！请特别留意的是，那个 /bin/kill -HUP ... 的意义，这一行的目的在于将系统的 syslogd 重新以其参数档(syslog.conf)的资料读入一次！也可以想成是 reload 的意思啦！ 由于我们建立了一个新的空的纪录文件，如果不执行此一行来重新启动服务的话， 那么记录的时候将会发生错误呦！！ (请回到资源管理的章节读一下 kill 后面的 signal 的内容说明)！

实际测试 logrotate 的动作

好了，设定完成之后，我们来测试看看这样的设定是否可行呢？给他执行底下的指令：

```
[root@linux ~]# logrotate [-vf] logfile
参数：
-v  ：启动显示模式，会显示 logrotate 运作的过程喔！
-f  ：不论是否符合设定文件的数据，强制每个登录档都进行 rotate 的动作！
范例：

范例一：执行一次 logrotate 看看整个流程为何？
[root@linux ~]# logrotate -v /etc/logrotate.conf
reading config file /etc/logrotate.conf
including /etc/logrotate.d
reading config file acpid
reading config info for /var/log/acpid
.....(中间省略).....
rotating pattern: /var/log/btmp  monthly (1 rotations)
empty log files are rotated, old logs are removed
considering log /var/log/btmp
  log does not need rotating

范例二：强制进行 logrotate 的动作
[root@linux ~]# logrotate -vf /etc/logrotate.conf
.....(前面省略).....
rotating pattern: /var/log/wtmp  forced from command line (1 rotations)
empty log files are rotated, old logs are removed
considering log /var/log/wtmp
  log needs rotating
rotating log /var/log/wtmp, log->rotateCount is 1
renaming /var/log/wtmp.1 to /var/log/wtmp.2 (rotatecount 1, logstart 1, i 1),
renaming /var/log/wtmp.0 to /var/log/wtmp.1 (rotatecount 1, logstart 1, i 0),
old log /var/log/wtmp.0 does not exist
renaming /var/log/wtmp to /var/log/wtmp.1
creating new log mode = 0664 uid = 0 gid = 22
```

```
removing old log /var/log/wtmp.2
# 看到否？整个 rotate 的动作就是这样一步一步进行的～
```

上面那个 -f 具有『强制执行』的意思，如果一切的设定都没有问题的话，那么理论上，您的 /var/log 这个目录就会起变化啰！而且应该不会出现错误讯息才对！嘿嘿！这样就 OK 了！很棒不是吗？！

好了，那么预设的 logrotate 什么时候执行呢？呵呵！不用担心，系统已经帮我们设定好了！放在哪里呢？刚刚不是提过吗？就是放在 /etc/cron.daily/logrotate 里面啊！如果您的版本不是 FC4 ， 那么可以利用 rpm 的相关功能找到这个设定数据喔！ ^_^

由于 logrotate 的工作已经加入 crontab 里头了！所以现在每天系统都会自动的给他查看 logrotate 啰！不用担心的啦！！只是要注意一下那个 /var/log/messages 里头是否常常有类似底下的字眼：

    Oct 24 15:15:35 localhost syslogd 1.4.1: restart.

这说明的是 syslogd 重新启动的时间啦 (就是因为 /etc/logrotate.d/syslog 的设定之缘故！) 底下我们来进行一些例题的练习，让您更详细的了解 logrotate 的功用啊！

假设前提是这样的，前一小节当中，假设您已经建立了 /var/log/admin.log 这个档案， 现在，您想要将该档案加上 +a 这个隐藏标签，而且设定底下的相关信息：

- 登录档轮替一个月进行一次；
- 该登录档若大于 10MB 时，则主动进行轮替，不需要考虑一个月的期限；
- 保存五个备份文件；
- 备份文件不要压缩

那你可以怎么样设定呢？呵呵～很简单啊！看看底下的动作吧！

```
# 1. 先建立 +a 这个属性啊！
[root@linux ~]# chattr +a /var/log/admin.log
[root@linux ~]# lsattr /var/log/admin.log
-----a------- /var/log/admin.log
[root@linux ~]# mv /var/log/admin.log /var/log/admin.log.1
mv: cannot move '/var/log/admin.log' to '/var/log/admin.log.1':permission deny

# 2. 开始建立 logrotate 的设定档，增加一个档案在 /etc/logrotate.d 内就对了！
[root@linux ~]# vi /etc/logrotate.d/admin
# This configuration is from VBird 2005/10/24
/var/log/admin.log {
        monthly
        size=10M
        rotate 5
        nocompress
        sharedscripts
        prerotate
                /usr/bin/chattr -a /var/log/admin.log
        endscript
```

```
        sharedscripts
        postrotate
                /usr/bin/killall -HUP syslogd
                /usr/bin/chattr +a /var/log/admin.log
        endscript
}

# 3. 测试一下 logrotate 相关功能的信息显示：
[root@linux ~]# logrotate -v /etc/logrotate.conf
.....(前面省略).....
rotating pattern: /var/log/admin.log  10485760 bytes (5 rotations)
empty log files are rotated, old logs are removed
considering log /var/log/admin.log
  log does not need rotating
not running shared prerotate script, since no logs will be rotated
.....(底下省略).....

# 4. 测试一下强制 logrotate 与相关功能的信息显示：
[root@linux ~]# logrotate -vf /etc/logrotate.d/admin
reading config file /etc/logrotate.d/admin
reading config info for /var/log/admin.log

Handling 1 logs

rotating pattern: /var/log/admin.log  forced from command line (5 rotations)
empty log files are rotated, old logs are removed
considering log /var/log/admin.log
  log needs rotating
running shared prerotate script
rotating log /var/log/admin.log, log->rotateCount is 5
....(中间省略)....
renaming /var/log/admin.log.0 to /var/log/admin.log.1 (rotatecount 5, logstart 1, i 0),
old log /var/log/admin.log.0 does not exist
log /var/log/admin.log.6 doesn't exist -- won't try to dispose of it
renaming /var/log/admin.log to /var/log/admin.log.1
running shared postrotate script

[root@linux ~]# lsattr /var/log/admin.log*
-----a------- /var/log/admin.log
------------- /var/log/admin.log.1
```

看到了吗？透过这个方式，我们可以建立起属于自己的 logrotate 设定档案， 很简便吧！尤其是要注意的， /etc/syslog.conf 与 /etc/logrotate.d/* 档案常常要搭配起来，例如刚刚我们提到的两个案例中所建立的 /var/log/admin.log 就是一个很好的例子～建立后，还要使用 logrotate 来轮替啊！ ^_^

分析登录档

登录档的分析是很重要的！例如那个 last 可以让你知道到底谁登录进主机啦！但是并没有 pop3 这个收信协议的登录讯息！这个时候就需要考虑到 /var/log/secure 的纪录啦！无论如何，既然系统有给我们几个指令可以约略观察，那就来查一查吧！ ^_^

一些常见指令： last, lastlog, dmesg

那就来谈一谈一些系统有的观察指令啊：

- dmesg

```
[root@linux ~]# dmesg | more
```

在指令列模式直接输入 dmesg 即可执行！由于系统在开机的过程当中尚未将硬盘 mount 上来，所以无法直接将数据直接给他读到 log file 当中去，但是为了除错上面的方便， 所以在开机的过程当中的讯息还是要记录下来，这个时候系统就将 ram 开了一个小区块来储存这个数据啰！这个开机记录的档案就是：『/proc/kmsg』啦！同时，预设的 RAM 的区块容量在不同的版本中并不相同，目前的预设版本是 16KB 的大小呦。

基本上，几乎所有的核心信息都可以使用 dmesg 来查阅得到的，举例来说， 想要知道开机有没有捉到网络卡，用的就是『 dmesg | grep 'eth' 』之类的指令啊！

- last

```
[root@linux ~]# last -n number
[root@linux ~]# last -f filename
参数：
-n  ：我们知道 last 会读出这个月的数据，若数据量太大，可使用 -n 来严格要求
      所以显示的笔数即可。例如 20 笔数据： last -n 20 或 last -20 均可。
-f  ：last 预设读出 /var/log/wtmp 这个档案，但是我们可以透过 -f 读取不同的
      登录文件信息喔！
范例：

范例一：将上个月的资料读出，仅读出 5 笔资料时？
[root@linux ~]# last -n 5 -f /var/log/wtmp.1
dmtsai2  pts/2           Mon Oct 24 14:18 - 14:18  (00:00)
dmtsai2  work:0 work     Mon Oct 24 14:18   gone - no logout
dmtsai2  work:0 work     Mon Oct 24 14:18 - 14:18  (00:00)
dmtsai2  pts/2           Mon Oct 24 14:18 - 14:18  (00:00)
dmtsai2  work:0 work     Mon Oct 24 14:18 - 14:18  (00:00)
```

够炫吧？！

- lastlog

```
[root@linux ~]# lastlog
```

```
Username  Port     From             Latest
root      tty1                      Tue Aug 16 18:06:20 +0800 2005
bin                                 **Never logged in**
.....(中间省略).....
dmtsai2   vbird-wo vbird-work       Mon Oct 24 14:18:49 +0800 2005
```

说穿了，lastlog 只是读出 /var/log/lastlog 内的信息而已～ 他会显示目前系统上面的所有账号当中，每个账号最近一次登入的时间喔！

鸟哥自己写的登录档分析工具：

虽然有一些有用的系统指令，不过，要了解系统的状态，还是得要分析整个登录档才行～ 事实上，目前已经有相当多的登录档分析工具，例如 FC4 上面预设的 logwatch 这个套件所提供的分析工具， 他会每天分析一次登录档案，并且将数据以 email 的格式寄送给 root 呢！ 你也可以直接到 logwatch 的官方网站上面看看：

- http://www2.logwatch.org:8080/

如果是非 FC4 系列的其它 distributions ，可以查阅一下您自己的 distributions 所提供的分析工具，如果没有，那么可以自行安装 logwatch 帮您分析啊～

虽然已经有了这些工具，但是鸟哥自己想要分析的数据毕竟与对方不同～ 所以啰，鸟哥就自己写了一支小程序 (shell script 的语法) 用来分析自己的登录文件， 这支程序分析的登录文件数据其实是固定的，包括有：

- /var/log/secure
- /var/log/messages
- /var/log/maillog

当然啦，还不只这些啦，包括各个主要常见的服务，如 pop3, mail, ftp, su 等会使用到 pam 的服务， 都可以透过鸟哥写的这个小程序来分析与处理呢～整个数据还会输出一些系统信息， 所以输出结果有点像底下这样：

```
##########################################################
欢迎使用本程序来查验您的登录档
本程序目前版本为： Version 0.1-1
程序最后更新日期为： 2005-01-09
若在您的系统中发现本程序有问题， 欢迎与我联络！
鸟哥的首页 http://linux.vbird.org
问题回报： http://phorum.vbird.org/viewtopic.php?t=3425
##########################################################


============== 系统汇整 =================================
核心版本  : Linux version 2.6.12-1.1456_FC4 (bhcompile@tweety.build.redhat.com)
CPU 信息  : Pentium III (Coppermine)
```

```
         : 997.519 MHz
主机名称   : localhost.localdomain
统计日期   : 2005/October/24 00:10:02 ( Monday )
分析的日期: Oct 23
已开机期间: 26 days, 1:02,
目前主机挂载的 partitions
      Filesystem            Size  Used Avail Use% Mounted on
      /dev/hda1             5.7G  3.3G  2.2G  61% /
      /dev/shm              189M     0  189M   0% /dev/shm


================= Ports 的相关分析信息 =======================
主机启用的 port 与相关的 process owner:
仅对本机界面开放的 ports (PID|owner|command)
      tcp 25|(root)|/usr/libexec/postfix/master
对外部接口开放的 ports (PID|owner|command)
      tcp 22|(root)|/usr/sbin/sshd
      udp 68|(root)|/sbin/dhclient -1 -q -cf /etc/dhclient-eth1.conf -lf /var/l

================= SSH 的登录文件信息汇整 =======================
一共成功登入的次数:           3
      账号        来源地址         次数
      dmtsai2     192.168.1.51       3

以 su 转换成 root 的使用者及次数
      账号                         次数
      (uid=530)                     1



================= POP3 的登录文件信息汇整 ======================
今日没有使用 POP3 的纪录



================= Postfix 的登录文件信息汇整 ===================
使用者信箱受信次数:
```

如果您有兴趣看看这支程序的话，欢迎下载：

- http://linux.vbird.org/download/index.php?action=detail&fileid=69

安装的方法也很简单，只要将上述档按下载后，解压缩，就会得到一个名为 logfile 的目录， 将此目录移动到 /usr/local/virus 当中，修改一下： /usr/local/virus/logfile.sh 档案， 里面的 email 与相关的信息只要修改一下，您就可以使用啦～ 啊！还要记得，将这支程序的执行写入 /etc/crontab 当中喔！可以在每天的 12:10am 执行这支小程序啦！ ^_^

本章习题练习

（ 要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看 ）

Linux 备份策略

最近更新日期：2005/10/25

万一不幸你的 Linux 被黑客入侵了、或是你的 Linux 系统由于硬件关系 (不论是天灾还是人祸) 而挂掉了！这个时候，请问如何快速的回复你的系统呢？呵呵！当然啰，如果有备份数据的话， 那么回复系统所花费的时间与成本将降低相当的多！平时最好就养成备份的习惯， 以免突然之间的手足无措！此外，哪些档案最需要备份呢？又，备份是需要完整的备份还是仅备份重要数据即可？ 嗯！确实需要考虑看看呦！

谁需要备份数据

前面的章节我们提到了相当多的 Linux 系统基础，这个时候我们再来谈一谈：『 若您的系统由于不预期的伤害，导致系统发生错误，该如何修复？ 』这可是个相当大的问题呀！怎么说呢？又什么叫做『不预期的伤害？』 好了，假如您的 Linux 系统上面某些 Internet 的服务套件是最新的！也意味着可能是『相对最安全的』， 但是，这个世界目前是闲人相当的多的，你不知道什么时候会有所谓的『黑客软件』被提供出来，万一你在 Internet 上面的服务套件被入侵，导致你的 Linux 系统全毁，这个时候怎么办？！

『重新安装就好啦！』或许您会这么说，但是，像鸟哥管理的几个网站的数据，尤其是 MySQL 数据库的数据，这些都是弥足珍贵的经验资料，万一被损毁而就不回来的时候，不是很可惜吗？ 这个还好哩，万一您是某家银行的话，呵呵！那么数据的损毁可就不是能够等闲视之的！！ 关系的可是数千甚至上万人的身家财产！！这就是备份的重要性了！他可以最起码的稍微保障我们的数据有另外一份 copy 的备援以达到『安全回复』的基本要求！

如果是针对个人数据的话，那么在一般桌上型计算机中，Norton 的『 Ghost 』应该算是一套好到不行的备份大师了！ 最主要是 Ghost 可以针对整个 partition 来进行备份，所以啰，我们可以将 Windows 系统当中的整个 C 或者是整个 D 槽完整的备份下来。甚至在还原方面也是非常的快速，而且操作简便！ 另外，由于个人桌上型计算机所使用的数据量通常不大，所以当 ghost 完成之后，通常只要将数据烧录到光盘片当中，大约只要一至两片的光盘片也就绰绰有余啰！ 那么将光盘片保存好，这就是最简易的数据备份模式啰！ 此外，由于个人的数据变动性不大，所以数据的备份频率方面也不需要非常的频繁！

但是，万一您的主机有提供 Internet 方面的服务呢？又该如何备份啊？ 举个例子来说，像是我们 Study Area 团队的讨论区网站 http://phorum.study-area.org 提供的是类似 BBS 的讨论文章， 虽然数据量不大，但是由于讨论区的文件是天天在增加的，每天都有相当多的信息流入， 由于某些信息都是属于重要的

人物之留言，这个时候，我们能够让机器死掉吗？

再提到 2002 年左右鸟哥的讨论区曾经挂点的问题，以及 2003 年初 Study-Area 讨论区挂点的问题，讨论区一旦挂点的话，该数据库内容如果损毁到无法救回来， 嘿嘿！要晓得讨论区可不是一个人的心血耶！有的时候（像 Study-Area 讨论区）是一群热心 Linux 的朋友们互相建立交流起来的数据流通网，如果死掉了， 那么不是让这些热血青年的热情付之一炬了吗？！所以啰，建立备份的策略是相当的重要的。

基本上，『计算机是一个相当不可靠的机器』这句话在大部分的时间内还是成立的！ 常常会听到说『要计算机正常的工作，最重要的是要去拜拜！』嘿嘿！不要笑！这还是真的哩！ 尤其是在目前一些计算机周边硬件的生产良率 (就是将硬件产生出来之后，经过测试， 发现可正常工作的与不能正常工作的硬件总数之比值) 越来越差的情况之下，计算机的不稳定状态实在是越来越严重了！

举个例子来说，鸟哥曾经同时买过同一厂牌的 IxM 30GB 硬盘三颗，回来之后经过一个星期，嘿嘿！挂掉了两颗！其中一颗是有坏轨，另外一颗是『完全死掉』， 拿去公司要求修理，结果呢？嗯！店家直接拿了一棵新的给我，害我吓一跳，店家的工程师说 『唉呀！目前这个牌子的良率太差了，所以代理商为了怕麻烦，都会直接拿新的替换给我们啦！』 要晓得的是，当初那一颗完全死掉的硬盘，是我用来备份我的主机数据的.... 好在当时我将备份的资料放在三四个地方，还好....

所以啰！由于计算机 (尤其是目前的计算机，操作频率太高、硬件良率太差、使用者操作习惯不良、 『某些』操作系统的当机率太高....) 的稳定性较差，所以啰！备份的工作就越来越重要了！

那么一般我们在备份时考虑的因素有哪些呢？

1. 备份哪些档案：
   例如在账号管理那一篇当中，我们知道最起码主机的账号信息在 /etc/* 及 /home/* 等等档案都是重要的！
2. 选择什么备份的媒介：
   是可擦写光盘、另一棵硬盘、同一颗硬盘的不同 partition、还是使用网络备援系统？ 哪一种的速度最快，最便宜，可将数据保存最久？
3. 考虑备份的方式：
   是完全备份(类似 ghost)还是部分备份即可？
4. 备份的频率：
   例如 MySQL 数据库是否天天备份、若完全备份，需要多久进行一次？
5. 备份使用的工具为何：
   是 tar 还是 cpio 等等？

底下我们就来谈一谈这些问题的解决之道吧！ ^_^

哪些 Linux 数据具有备份的意义

具有备份意义的档案通常可以粗分为两大类，一类是系统、一类则是类似网络服务的数据， 那么各有哪些档案需要备份的呢？我们就来稍微分析一下。

- 主机系统需要备份的档案：

这方面的档案主要跟『账号与系统设定文件』有关系！ 主要有哪些账号的档案需要备份呢？就是 /etc/passwd, /etc/shadow, /etc/group, /etc/gshadow, /home 底下的使用者家目录等等， 而由于 Linux 预设的主机信息设定参数文件都在 /etc/ 底下，所以只要将这个档案备份下来的话， 那么几乎所有的设定档都会存在的！

至于 /home 底下是每个人的家目录，自然也需要来备份一番！再来，由于使用者会有邮件吧！所以呢，这个 /var/spool/mail 内容也需要备份呦！另外，由于如果您曾经自行更动过核心，那么 /boot 里头的信息也就很重要啰！所以啰，这方面的数据您必须要备份的档案为：

- /etc/ 整个目录
- /home 整个目录
- /var/spool/mail
- /boot
- /root
- 如果您自行安装过其它的套件，那么 /usr/local/ 或 /opt 也最好备份一下！

- 网络服务的数据库方面：

这部份的数据可就多而且复杂了，如果您的网络套件设定都是以原厂提供的为主，那么您的设定档案大多是在 /etc 底下，所以这个就没啥大问题了！但是若您的套件大多来自于自行的安装，那么 /usr/local 这个目录可就相当的重要了！这里我们假设我们提供的服务套件都是使用原厂的 RPM 安装的！所以要备份的数据文件有：

- 数据设定档案：
  /etc/ 整个目录 /usr/local/ 整个目录
- 系统 www + MySQL：
  WWW 资料：/var/www 整个目录或 /srv/www 整个目录，及系统的使用者家目录
  MySQL ： /var/lib/mysql 整个目录
- 其它你在 Linux 主机上面提供的数据库数据文件！

- 推荐需要备份的目录：

由上面的介绍来看的话，那么如果您的硬件或者是由于经费的关系而无法全部的数据都予以备份时， 鸟哥建议您至少需要备份这些目录呦！

- /boot
- /etc
- /home
- /root
- /usr/local(或者是 /opt 及 /srv 等)
- /var(注：这个目录当中有些暂存目录则可以不备份！)

- 不需要备份的目录：

有些数据是不需要备份的啦！例如我们在 档案权限与目录配置 里头提到的 /proc 这个目录是在记录目前系统上面正在跑的程序 (processes) ，这个数据根本就不需要记录的呢！所以就把他拿掉！此外，外挂的机器，例如 /mnt 或 /media 里面都是挂载了其它的硬盘装置、光驱、软盘机等等，这些也不需要备份吧！？ 所以啰！底下有些目录可以不需要备份啦！

- /dev ：这个随便你要不要备份
- /proc：这个真的不需要备份啦！
- /mnt ：根据版本不同，有的是 /media 如果你没有在这个目录内放置你自己系统的东西，也不需要备份
- /tmp ：干嘛存暂存档！不需要备份！

选择的备份装置

在备份的时候，选择一个『数据存放的地方』也是很需要考虑的一个因素！ 什么叫做数据存放的地方呢？讲个最简单的例子好了，我们知道说，较为大型的机器都会使用 tap 这一种磁带机来备份数据，而如果是一般个人计算机的话，很可能是使用类似 Mo 这一种可擦写式光盘片来存取数据！但是您不要忘记了几个重要的因素，那就是万一您的 Linux 主机被偷了呢？这不是不可能的，我们隔壁校区的研究室曾经遭小偷，里面所有的计算机都被偷走了！包括『 Mo 片』，当他们发现的时候，一开始以为是硬件被偷走了，还好，他们都有习惯进行备份，但是很不幸的，这一次连『备份的 MO 都被拿走了！』怎么办？！只能道德劝说小偷先生能够良心发现的将硬盘拿回来啰！唉～真惨....

这个时候，所谓的『远程备援系统』就显的相当的重要了！ 什么是远程备援呀！说的太文言了！呵！简单的说，就是将你的系统数据『备份』到其它的地方去， 例如说我的机器在台南，但是我还有另一部机器在高雄老家，这样的话，我可以将台南机器上面重要的数据都给他定期的自动的 ftp 回去！也可以将家里重要的数据给他丢到台南来！这样的最大优点是可以在台南的机器死掉的时候， 即使是遭小偷，也可以有一个『万一』的备份所在！但是缺点是～～频宽严重的不足！ 在这种状态下，所能采取的策略大概就是『仅将最重要的数据给他 ftp 回去啰！』至于一些只要系统从新安装就可以回复的咚咚！那就没有这个必要了！当然啰，如果你的网络是属于 T1 专线的话，那么完整备份将数据丢到另一地去，呵呵！也是很可行的啦！只是我没有那么好命....唉～穷人一个～

在此同时，我们再来谈一谈，那么除了这个『相对较为安全的备份』方法之外， 毕竟这种网络备援系统实在是太耗频宽了！如果像我们一般家用的 ADSL 根本就是吃不消！那么怎么办！还有其它的方法吗？喔～那就只好使用近端的装置来备份啰！ 这也是目前我们最常见到的备份方法！例如一般我们使用的 Tape, Mo, Zip, CD-RW, DVD-RW 还有备份用抽取式硬盘与携带式硬盘等等！ 那么在选择上需要注意些什么呢？需要注意的地方有几点：

- 速度要求：

『备份』基本上在 Linux 主机上面也是蛮耗系统资源的！因为需要将系统的数据拷贝到其它装置上面去，这个时候 CPU 几乎是 loading 100%！您总不希望系统就这样给他挂点吧！？此外，有些系统的数据实在太多咯， 怎么样也备份不完！所以啰，越快的储存装置是越好的！如果您是个重视速度甚于一切的人， 那么我觉得抽取式硬盘是个不错的方式，只不过.....目前我知道的抽取式硬盘都需要冷开机才行，不太符合 Linux 主机 24 小时全年无休的状态....

但是硬盘真的越来越大、越来越便宜了，不使用速度快的硬盘来备份实在很可惜～ 加上目前的火线 (IEEE 1394) 以及 USB 2.0 外接式硬盘盒技术已经相当的成熟， 传输速度又快，又可以直接热拔插 (Plug and Play)，接上 USB 硬盘， 整个复制一下，传输速度理论上可达 480Mbps (约 60 MBytes/second) ，快的哩！ 复制完毕，又可以将硬盘带走，不需要与主机放置在一起，还可以避免同时被偷，真是不错。

但是，硬盘还是有一定的困扰，那就是『不接电源的硬盘需要很好很好的保养』。 我们知道计算机最好的

保养就是常常开机去运作一下，免得长期不开机，造成受潮而损坏。 这个携带式硬盘只是偶而才会连上主机来进行备份的数据，除非您额外购买一部防潮箱来放置硬盘， 否则很容易损坏ㄟ！所以，近年来速度越来越快的 DVD-RW 就变的很方便啰！至于 tap ，在速度上完全是落后的.....至于使用第二颗硬盘备份，类似 Raid, 或者是安装一颗备份的硬盘在 Linux 系统当中，这个方案也很好， 而且速度上绝对是最具优势的！但是就如同我们刚刚提到的，万一你的机器被偷了，连带的， 这颗备份的硬盘自然也就不见了.....

- 储存容量：

这也是一个需要考虑的因素！而且常常是最大考虑的因素呢！虽然目前硬盘越来越便宜， 但是毕竟就如同前面说的，抽取式硬盘需要将系统冷开机，而建构在系统内的硬盘又同时具由不安全的成分在， 携带式硬盘可能又有不容易保存的特性，这个时候一个大容量的替代方案就显的很重要了！虽然 CD-RW 与 DVD-RW 可以提供不错的速度，但是其容量毕竟不足 (目前的 DVD-RW 片最大虽然可以突破 8.7GB ，但是，贵的很哩～期待新规格赶紧定义出来呢！)， 所以说，具有大容量的 tap (磁带容量最小的一款也可以到达 8 GB 左右！) 就相当的具有这方面的优势了！而且携带方便，存放也容易！更可以带着走～～

- 经费与资料可靠性：

在经费不短缺的情况下，我们当然会建议您上面的几个装置都买一买， 然后分别在不同的时间进行不同的备份作业（底下我们有些建议的啦！ ^_^）！但是如果经费也是需要考虑的话，那么磁带机这个目前还算贵重的物品可能暂时还动不到！ 这个时候近来渐渐便宜的 DVD-RW 就显的活跃的多了！而且光盘片也可以保存很久的ㄋㄟ～ 当然，目前应该不会有人以软盘来备份了吧！？呵呵！软盘可是相当不安全的 （每次我看到有人拿软盘拷贝数据，我都会要他 copy 完成之后，立刻到另外一部计算机 copy 出来试看看，果不其然，十次里面有八次对方的软盘片都有问题～）

无论如何，如果经费够的话， Tape 备份数据真的是一个不错的点子！因为他的高容量让我好满意！ 再来，如果经费稍微短缺的话，那么 DVD-RW 经常性的将数据烧录下来，这也是蛮好的，尤其 DVD 片又不占空间！ 再来，如果还是没有办法，那么一颗内建在 Linux 的硬盘用来备份也是不错的！什么！！连备份的硬盘都没有，唉！怎么跟我一样～ 这个时候没办法啦，用原来的安装系统的硬盘，多留一个 partition 用来当作备份之用吧（这也是目前鸟哥常用的方法之一！）底下我们来看一看一些常见的装置代号！

- 光驱： /dev/cdrom
- 磁带机： /dev/st0 (SCSI 界面), /dev/ht0 (IDE 界面)
- 软盘机： /dev/fd0, /dev/fd1
- 硬盘机： /dev/hd[a-d][1-16] (IDE 界面), /dev/sd[a-p][1-16] (SCSI 界面)
- 抽取式 USB 规格硬盘机： /dev/sd[a-p][1-16] (别怀疑，刚好与 SCSI 接口相同！)
- 打印机： /dev/lp[0-2]

特别留意的是磁带机呦!如果你有钱的话,那么买一部磁带机是相当不错的建议!没钱的话,买 IDE 或 SATA 接口的硬盘也很不错！！ ^_^

备份的种类

讲了好多口水了，还是没有讲到重点，真是的....好了，再来提到那个备份的种类， 其实前面已经提到一些了！基本上，备份就可以直接分为『完全备份』与『部分备份』这两方面：

- 完全备份(Full Backup)：

完全备份就是将根目录『 / 』里头所有的数据都给他一股脑儿的备份下来，不过， 这个时候所需要的『时间与备份装置的容量』 就显的相当的重要了！用在大型的企业是有一定的需求的，但是像我们这一种小网站的话， 完整的备份似乎太过于浪费的（毕竟我们可以用时间换取金钱....了不起又重新安装了....）。

但是刚刚完成的系统（还没有对外 Internet 上面服务）通常可以的话，就赶快给他备份一下吧！ 这样的一个备份是最干净的！用在系统的最干净还原是相当有帮助的！这是因为有的时候我们的系统被入侵了，但是 root 并不知道，这个时候老是拿最近的备份数据来还原也没有用呀！因为连同被 cracker 修改过的档案也被我们备份下来了呀！呵呵！所以啰， 将一个最原始的系统的数据备份下来还是有其必要性的！此外， 这种完整备份的频率可不能太高，因为太耗系统资源了！

- 部分备份(Implement backup)

部分备份就如同上面提过的，备份那些最重要的数据就好了！反正系统不见的话， 只要重新安装就回来了，数据只要妥善的备份重点数据，那么系统的复原还是一个很快速的工作！ 以鸟哥为例，我通常都喜欢仅备份最重要的信息，因为重新安装一次系统时间花的并不长 （一个钟头内一定可以搞定！）而账号、服务设定、原本系统的数据库、等等，几乎都只要 copy 回来就 OK 了！

例如：我们这个网站在开始营运初期，虽然交通流量很大了，还是常出状况， 那个时候重新安装了好几次（似乎是硬件的问题！），每次都是一个下午就搞定了！所以啰， 鸟哥是一直认为『重点备份』就真的是蛮重要的！ 尤其很多时候，你的数据被吃掉都只是『某个网络服务』， 那么如果仅进行完整备份的工作时，呵呵！单单是将系统先读出来，再取出所要还原的部分， 呵呵！就可以累死你了....

如同上面提到的，这两个方式各有优缺点啦！那么如果可能的话，是否两个都来进行一下最好呢？呵呵！ 答对了！给你拍拍手！我们通常的规划就是这样，重点部分的备份频率较高，可能每天都需要备份的！ 至于如果是完整备份的话，那么一个星期、甚至一个月在备份一次都可以！目前鸟哥的系统上面就是每天备份 MySQL 数据库，然后每个星期备份所有的重要数据！

备份的工具选择

好了，选定了备份的装置与备份的频率之后，那么我要使用什么方式来备份呀！呵呵！ 这个也要跟备份的种类相互配合呢！通常鸟哥在备份的时候，除非有磁带机或者有特殊的功用，否则通常我只使用 tar 啦！但是这里我们介绍一下 cpio 这个东西！如果你有磁带机的话， cpio 可是相当好用的一个指令呢！ ^_^ ，呵呵！ 由于这两个指令我们早在『压缩工具』当中稍微提过了， 请再自行过去瞧一瞧去啰！另外，您或许会问道：『那么我为什么不直接给他 copy 过去备份的地点就好了呢！？』呵呵！既然可以在 copy 的过程里面增加压缩的功能来减低整个储存空间的消耗，为何不压缩？！那当然还是压缩一下比较好啰！ 所以啊！鸟哥还是比较喜欢 tar, cpio 的啦！

- 完整备份的工具：

在完整备份的工具里面，三个工具 tar, cpio 与 dump 都很常被使用！此外，那个 dd 也是不错的指令喔！至于 cpio 的话，他最大的好处就是『 cpio 连一般的装置文件都可以 copy 过来!』很棒吧!所以使用 cpio 进行完整备份是很棒的一个选择。不过需要注意的是，由于 cpio 需要配合 find 才可以正常的动作！这里请特别留意啰！另外，使用 cpio 常常配合另一颗完全用来备份的硬盘或者是磁带机才好！至于完整的指令用法请到 『压缩工具』 那一篇去查看啰！备份与反备份分别可以这样使用：

```
# 1. 使用 cpio 来备份与还原:
```

```
[root@linux ~]# find / -print | cpio -covB > /dev/st0   <==备份到磁带机
[root@linux ~]# cpio -iduv < /dev/st0                   <==还原

# 2. 使用 tar 来备份与还原
[root@linux ~]# tar --exclude /proc --exclude /mnt --exclude /tmp \
> -zcvpf host.tgz /
[root@linux ~]# tar -zxvf host.tgz

# 3. 用 dd 来备份一颗完全一模一样的硬盘：
[root@linux ~]# dd if=/dev/hda of=/dev/hdb
# 完整的将 /dev/hda 通通备份到 /dev/hdb 当中去！
```

这几个工具都蛮好用的！尤其鸟哥特喜欢 tar 的用途！因为他相当的适合于另一颗硬盘的备份呢！ 当然，如果您有完全相同的两颗硬盘时，用 dd 会是一个很不错的完整备份的方案喔！ ^_^

- 部分备份的基础工具：

至于部分备份方面，我们就以简单的 tar 来说明一下吧！！假如我们需要备份的数据是每天的 MySQL 数据库时，由于我想让每天的数据都存成不同的档案，而要分别档案的新旧又以日期来分别最简单了！ 所以我就可以这样做：

```
[root@linux ~]# tar -zpcvf mysql.`date +%Y-%m-%d`.tgz /var/lib/mysql
# 如果忘记了上面的指令代表什么意思，那么请回到 bash 那一章去瞧一瞧吧！

[root@linux ~]# tar -N '2005/10/25' -zpcvf home.tgz /home
只有在比 2005/10/25 还要新的档案，在 /home 底下的档案才会被打包进 home.tgz 中！
```

这样就能将 mysql 的数据库压缩备份至 mysql.2005-10-25.tgz 这个档案， 并且日期会每天都不一样！呵呵！这样一来如果系统的数据库出了问题，就可以马上的回复了！ 而且还有很多的档案可供回复呢！不错吧！此外，也可以利用类似上面的第二个范例的例子， 将最新的资料备份就好，其它的资料则不予以更动！嘿嘿！提供了更完善的方式呢！

大致的工具就是这样了！此外，由于备份是长长久久的事业，所以我们需要的是『系统可以自己动作』的方式， 您说是吧！所以呢，这个时候就需要使用到 cron 的服务啦！还记得我们先前讲过的 例行性命令的建立 吗？赶快再去复习一下呀！

鸟哥的备份策略

其实鸟哥在备份的策略相当的简单，我并没有想要将整个系统完全的备份下来，因为太耗时间了！ 而且就我的立场而言，似乎也没有这个必要，所以通常鸟哥只备份较为重要的档案而已！不过，由于我需要备份 /home 与网页数据，如果天天都备份，我想，系统迟早会受不了 (因为这两个部分就已经快要占去 1GB 的硬盘空间...) ，所以鸟哥就将我的备份分为两大部分，一个是每周备份，一个则是每日备份，备份的时间点都选择在凌晨的 3~4 点左右！这个时候我就写了两个简单的 scripts ，分别来储存我的数据。所以针对鸟哥的网站 (简称『鸟站』哈哈！)我的备份策略是这样的：

1. 使用一颗加挂的硬盘来进行备份的功能，挂在 /disk2 当中；
2. 每周进行的备份有 /home, /var, /etc, /boot, /usr/local 等目录；

3. 每日进行的目前仅有 MySQL 数据库；
4. 利用 /etc/crontab 来自动提供备份的功能；
5. 在每周或每月定期的将数据分别 (a)烧录到光盘上面 (b)使用网络传输到另一部机器上面。

那就来看看鸟哥是怎么备份的吧！ ^_^

日常备份行为：

底下提供鸟哥的备份的 scripts ，希望对大家有点帮助！ 我的动作是：1) 先将所有的数据通通丢到 /disk2/backup 底下去，然后 2) 进行压缩打包，并且传送到内部的 192.168.1.100 那部主机上面去。

```
# 1. 每周备份的资料的 script 啊！
[root@linux ~]# mkdir /disk2/backup
[root@linux ~]# vi /disk2/backup/backupweekly.sh
#!/bin/bash
# =========================================================
# 说明：
# 这支程序是用来备份鸟哥的网站资料的！ 当然啦，数据量很大的！
# 我将他分为几个部分：
#       第一部份是系统的服务与受服务的设定档，
#       第二部分则是与使用者有关的重要信息部分了！ ^_^
# =========================================================
# History
# When          Who     What
# 2000/12/16    VBird   first time to release
# 2002/03/26    VBird   Adding ftp services' backup in /disk2/backup/ftp
# 2003/07/03    VBird   发现解压缩之后会有一些错误数据发生！
#                       所以将 tar 加入 -p 的参数！
# 2005/01/02    VBird   怀疑可能因为备份期间硬盘运转的问题导致当机，
#                       所以，加上多个 sleep 以及 sync 的功能！
# =========================================================
PATH=/bin:/usr/bin:/sbin:/usr/sbin; export PATH
LANG=C; export LANG
LC_ALL=C; export LC_ALL

# 设定路径来备份
basedir=/disk2/backup
named=$basedir/named
postfixd=$basedir/postfix
vsftpd=$basedir/vsftp
sshd=$basedir/ssh
sambad=$basedir/samba
wwwd=$basedir/www
others=$basedir/others
```

```
userinfod=$basedir/userinfo

# ================================================================
# 1.  系统的相关服务，主要服务有：
#     (1) BIND server:
#     (2) Postfix:
#     (3) vsftp:
#     (4) sshd:
#     (5) samba:
#     (6) WWW:
#     (7) Others:其它系统必须要的一些信息!

# 1.1 系统的 BIND 套件，主要是 DNS 的设定档备份!
cp -a /var/named/chroot/etc     $named/chroot
cp -a /var/named/chroot/var     $named/chroot

# 1.2 系统的 Postfix Server 相关的档案备份数据!
cp -a /etc/postfix/*            $postfixd 2> /dev/null
cp -a /etc/rc.d/init.d/postfix  $postfixd
cp -a /etc/dovecot.conf         $postfixd

# 1.3 系统的 vsftpd 服务器的仅有的设定档喔!
cp -a /etc/vsftpd/*             $vsftpd
cp -a /etc/vsftpd.*             $vsftpd

# 1.4 系统的 sshd 服务器的设定档案!
cp -a /etc/ssh/*                $sshd

# 1.5 系统的 Samba 所动用的档案
cp -a /etc/samba/*              $sambad

# 1.6 WWW
cp -a /etc/my.cnf                       $wwwd
cp -a /etc/php.ini                      $wwwd
cp -a /etc/httpd/conf/httpd.conf        $wwwd
cp -a /etc/httpd/conf.d                 $wwwd
cp -a /etc/httpd/conf.d/vbird.conf*     $wwwd
cd /usr/local
   tar -pcf $wwwd/counter-data.tar  Counter/data
cd /var/lib
   tar -pcf $wwwd/mysql-lib.tar  mysql --exclude mysql/mysql.sock
cd /var
   tar -pcf $wwwd/www-cgi-icon.tar  www/cgi-bin www/icons
```

```
# 1.7 Others
cp -a /etc/hosts              $others
cp -a /etc/hosts.allow        $others
cp -a /etc/hosts.deny         $others
cp -a /etc/modprobe.conf*     $others
cp -a /etc/fstab              $others
cp -a /etc/resolv.conf        $others
cp -a /etc/shells             $others
cp -a /etc/wgetrc             $others
cp -a /etc/crontab            $others
cp -a /etc/sysconfig/i18n     $others
cp -a /etc/sysconfig/network  $others
cp -a /etc/sysconfig/network-scripts/ifcfg-eth0 $others
cd /
   tar -pcf $others/etc.tar etc
cd /usr
   tar -pcf $others/local.tar local

sleep 5s
sync; sync

# ============================================================================
# 2.  主机的重要数据与数据库系统
#     (1) 使用者的信息  重点在 /etc/passwd, shadow, group 以及电子邮件、家目录

# 2.1
cp -a /etc/passwd  $userinfod
cp -a /etc/shadow  $userinfod
cp -a /etc/group   $userinfod

cd /var/spool
   tar -pcf $userinfod/mail.tar mail

cd /
   tar -pcf $userinfod/home.tar home --exclude home/lost+found

sleep 5s
sync; sync

# ============================================================================
# 3. 将主机的重要数据复制到 192.168.1.100 那部机器上面去!
```

```
# 3.1 压缩与打包
cd $basedir
   tar -zpcf backupweekly.tar.gz * --exclude backupweekly.tar.gz

sleep 5s
sync; sync

# 3.2 ftp 到 192.168.1.100
id="username"
pw='yourpassword'
cd $basedir
ftp -n 192.168.1.100 > $basedir/backup.ftp.log 2>&1 <<EOC
user    $id     $pw
binary
cd      /disk2/backup/
put     backupweekly.tar.gz
bye
EOC
sync; sync
```

当然啰，上面的 script 是适合鸟哥的状态，所以，你要使用的话，还得要修修改改哟！ 不要照着使用，会有问题的！另外，上面的 script 当中，我已经加上了远程储存的功能了， 那就是 #3.2 的 FTP 部分，藉由这个简单的动作，就可以将我这一部机器上面的数据， 整个传送到 192.168.1.100 那部机器上面，够简单吧！ ^_^

每日备份资料 scripts：

再来，继续提供一下每日备份的数据：

```
# 提供的是每日备份的 script 啊
[root@linux ~]# vi /disk2/backup/backupdaily.sh
#!/bin/bash
#
# This program is created by VBird 2002/06/13
#
# What is this program?
#       This program will backup the following messages:
#       1. MySQL data files ( /var/lib/mysql );
#       2. HTTP's CGI-directory ( /var/www/cgi-bin )
#
# HOW TO RUN THIS PROGRAM?
#       Just put the file into /etc/crontab job,
#       or put this file's link file to /etc/cron.daily!
#
```

```
##############################################################
# History
#  Date          What                                    Who
#=============================================================
# 2002/06/13     First time to run this program
#                The only backup files are MySQL and CGI VBird
#=============================================================
# 0. Get the date messages and backup directory
day=`date +%Y-%m-%d`
basedir="/disk2/backup/daily"

# 1. MySQL  ( PATH = /var/lib/mysql )
/etc/rc.d/init.d/mysqld stop
cd /var/lib
tar -zcf "$basedir"/mysql."$day".tar.gz mysql 2> /dev/null
/etc/rc.d/init.d/mysqld start

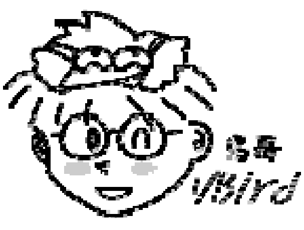
# 2. CGI ( PATH = /var/www/cgi-bin )
cd /var/www
tar -zcf "$basedir"/cgi-bin."$day".tar.gz cgi-bin 2> /dev/null
```

好啦！这样一来每天的 MySQL 数据库就可以自动的被记录在 /disk2/backup/daily 里头啦！而且还是文件名称会自动的改变的呦！呵呵！我很喜欢！OK！再来就是开始让系统自己跑啦！ 怎么跑？！就是 /etc/crontab 呀！提供一下我的相关设定呦！

```
[root@linux ~]# vi /etc/crontab
# 加入这两行即可 ( 请注意您的档案目录！不要照抄呦！ )
# backup scripts
30 3 * * 0 root /disk2/backup/backupweekly.sh
30 2 * * * root /disk2/backup/backupdaily.sh
```

这样系统就会自动的在每天的 2:30 进行 MySQL 的备份，而在每个星期日的 3:30 进行重要档案的备份！呵呵！您说，是不是很容易呢！？但是请千万记得呦！还要将 /disk2 当中的资料 copy 出来才行耶！否则整部系统死掉的时候....那可不是闹着玩的！

Tips:
有些时候，您在进行备份时，被备份的档案可能同时间被其它的网络服务所修改喔！ 举例来说，当您备份 MySQL 数据库时，刚好有人利用您的数据库发表文章，此时， 可能会发生一些错误的讯息。要避免这类的问题时， 可以在备份前，将该服务先关掉，备份完成后，再启动该服务即可！ 感谢网页 duncanlo 提供这个方法！



远程备援系统：

除此之外，那么还有没有保险的方式呢？呵呵！刚刚前面不是提过远程备援吗？ 最简单的说法就是『使用因特网的方法，将你的数据送到远程主机去备份！』那样就 OK 啦！那么我们怎么使用远程备份的方法呢？

那就用最简易的 FTP 吧！详细的资料， 其实上面我们就已经提到的，您可以自行参考一下。这里仅提供一个简单的说明：

1. 备份的资料最好『越精简越好』；
2. 远程主机必需提供 FTP 服务(当然，其它的服务例如 sftp 也是可以的，只要能够将数据传上去就好了！)
3. 远程主机必需要可以符合你上传的属性设定，例如 quota 容量、储存目录的属性等等！

如果你想要有自己的简单的自动 FTP scripts 来帮我们达成自动档案上传的功能，那就这样吧！

```
#!/bin/bash
#
# WHAT IS THIS:
# This program will automatically put the backup file
# from this host to another hosts
#
# HISTORY
# When          Who     What
# 2002/10/14    VBird   First time to release
#
###################################################################
# 1. input your FTP's ID and PassWord
  host="192.168.1.100"
  id="testing"
  pw='your.passwd'

# 2. what is the correct and remote working directory
  basedir="/disk2/backup"              # 本机上面的欲上传档案路径
  remodir="/disk2/backup_testinghost"  # 远程主机欲备份的目录

# 3. the tar file
  filename="backup.testinghost.tar"

# 4. starting tar work
# 因为我将所有预备被传送的数据都分门别类的放置在
# /disk2/backup 这个目录中， 但是 FTP 传送档案的时候，
# 毕竟一个档案送完会比较容易与快速，所以我先将好几个档案 tar 成一个！
  cd $basedir
  tar -cvf $filename *

###################################################################
# 5. 底下就是 ftp 自动联机并操作上传手续的 script ！
  ftp -n "$host" > "$basedir"/"$filename".log 2>&1 <<EOC
  user    $id         $pw
  binary
```

```
cd      $remodir
put     $filename
bye
EOC
```

好了！大家赶紧写一个适合自己的备份 script 来进行备份的行为吧！重要重要喔！ 如果不想要使用 FTP 而想要利用其它方式的话，或许可以 man rsync 这个指令来达成喔！

如何还原系统的考虑

既然有备份，自然就得要考虑到还原啰～一般来说，我们针对备份与还原的考虑通常有这两种：

- 我的主机需要在『很短的时间内』立刻就能够还原上线；
- 我的主机希望能够在查询到挂点的原因后，才正式上线；

以第一种类来说，例如您的公司是提供数千甚至数万人连上来的企业， 那么贵公司的服务器是『没有休假的权力』的。所以啰，除非真的是天灾，否则只要系统发生问题， 您都应该要在很短的时间内将该伺服器重新 on-line 才行啊！ 这个时候，一部一模一样的备份机器，或者是一份一模一样的完整备份数据，就显的很重要！ 因为即使是被入侵，反正先上线，然后用那部被入侵的机器赶紧进行修复与还原的工作， 这样才能够达到公司正常营运的目的啊！

那如果仅是一般个体户呢？例如鸟哥的网站，就是这样的代表。 那么我当然不必要选择完整备份了，因为即使我完整备份了，但是备份之初的数据如果就已经被入侵， 那我还原一个被入侵的数据有什么好处？加上目前的硬件等级都很高， 网络频宽都很够的前提之下，重新安装最新版 Linux 与更新到最新的套件，其实速度上是很快的。 如此一来，反正系统如果挂了，刚好整个 distribution 一起进行升级～似乎也不错啊～ ^_^

总之，备份工具是死的，使用的您是活的，您在使用 tar, cpio, dd 等工具时， 必须要优先考虑您所想要达到的『目的』是什么，再根据这个目的，以及您所可以利用的『资源与经费』， 选择适当的备份方式与媒体，这样就足够啦！大家加油啦！ ^_^

- 关于储存架构的考虑

在这个章节当中，我们提到的大概都是属于中小企业环境所使用到的备份动作， 如果您所在的公司是大型企业的环境，那么不妨考虑外接式，或者是拥有完整备份储存网络的架构， 例如 SAN (Storage Area Network) 及 NAS (Network Attached Storage) 设备， 这样才能够更快速与高效率的储存与备份啊！^_^

不过，这两种架构当中，都需要比较高价位的光纤信道，或者是新技术的 iSCSI 协议的需求， 鸟哥还无缘碰到。除了 iSCSI 有接触到一部机器之外， SAN 还真是没有碰过..... 这两部分的企业应用，就有待您自行与相关的厂商接触了解啰！ ^_^

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 你所看到的常见的储存设备有哪些？

Floppy, Mo, Zip, CD-RW, DVD-RW, 外接式 USB 硬盘, Tape, 外接式储存数组 (RAID)，额外的储存架构，如 SAN, NAS 等。

X Window 简易设定介绍

最近更新日期：2005/07/11

有人说，要让 Linux 更被大众所接受，那么更具亲和力的 X Window 系统是势在必行的！ X Window System 的整体架构其实不很好理解，因为他又分为 X Server 与 X Client 等部分， 再加上很好用的 Window Manager 来凑一脚，嘿嘿！更难以理解了。 不过，我们这里并没有介绍很深入的 X Window 架构，只是简单的介绍一下， 您应该如何设定 X Window 好让您的桌面系统 (Desktop) 变的更漂亮～ 尤其是某些学术用软件，需要 3D 加速的功能时，理解这些咚咚，就很重要了！

什么是 X Window System

在 Unix Like 的系统当中，可以利用鼠标与键盘来进行图形化接口的操作，那就是 Graphical User Interface (简称 GUI)接口啦！而我们将这个图形接口称为 X Window System。 为什么称为 X 呢？因为就英文字母来看， X 是在 W(indow) 后面，因此， 早期人们就戏称这一版的窗口接口为 X 啰！

事实上， X Window System 不是很容易理解，尤其是我们还没有接触到网络主机服务器方面的介绍。 不过，无论如何，要对 X 有一些概念的话，还是得要介绍一下他的运作原理才行啊～伤脑筋～ 底下鸟哥将就 X Window 的相关知识作个简单的介绍啊！

我是否需要启用 X Window System

在开始之前，还是得就这个话题来说明说明。一般来说，如果您的 Linux 主机定位为 Network Servers 的话，那么由于 Linux 里面的主要的服务的设定文件都是 ASCII 纯文字的格式档案，相当的容易设定的，所以啊，根本就是不需要 X Window 存在呢！ 因为 X Window 仅是 Linux 系统内的一个软件而已啊！

但是万一您的 Linux 主机是用来作为您的 Desktop 桌上计算机用的， 嘿嘿！那么 X Window 对您而言，就是相当重要的一个咚咚了！因为我们日常使用的办公室软件， 都需要使用到 X Window 图形的功能呢！

此外，以鸟哥的例子来说， 我之前接触到的数值分析模式，需要利用图形处理软件来将数据读取出来， 所以在那部 Linux 主机上面，我一定需要 X Window 的。此外，由于处理的软件有两种， 其中一种需要使用到 3D 的加速功能。但 Linux 预设的显示卡驱动程序对 3D 的加速功能有限， 此时，俺就得要重新安装显示卡驱动程序呢！伤脑筋～

回归到主题上面，除了主机的用途决定您是否需要启用 X Window 之外， 主机的『配备』也是您必须要考虑的一项决定性因素。 因为 X Window 如果要美观，可能需要功能较为强大的 KDE 或 GNOME 等窗口管理员(Window Manager)的协助，但是这两个庞然大物对于系统的要求又很高， 除了 CPU 等级要够，RAM 要足之外，显示卡的等级也不能太差～ 所以，早期的主机可能对于 X Window 就没有办法具有很好的效率了。

也就是说，您如果想要玩 X Window 的话，特别需要考虑到这两点：

- 稳定性：X Window 仅是 Linux 上面的一个套件 (或者也可以称为服务)，您不能对 X Window 与 Linux 的整合有太高的期望的，虽然目前的 X window 已经整合得相当好了。此外，任何程序的设计都或多或少会有些臭虫， X 当然也不例外。因此，在您的 Linux server 上面启用 X 系统的话， 自然多一个服务的启用，就会产生一些不确定性。因此，不是很建议对 Internet 开放的服务器启动 X Window 的啦！

  Tips:
  鸟哥刚开始接触 Linux (大约是在 1999 年左右)时， 由于不熟，通常都是预设给他启用 X Window 在我的主机上面的。 不过，那个时候的图形接口与 Linux kernel 的整合度比较差， 老是挂点去，是常常造成我其它 Internet 上面的服务无法顺畅的原因之一呢！

-
- 效能： 无论怎么说，程序在跑总是需要系统资源的，所以，多启用了 X 就会造成一些系统资源的损耗。 此外，上面也稍稍提到，某些 X 的软件是相当耗费系统资源的呢！ 所以，多起动 X 就可能造成您的系统效能的低落哩！

X Window 发展历史

X Window 最早是由 MIT (Massachusetts Institute of Technology) 在 1984 年发展出来的， 他们称这个咚咚为 X 。在经过数年的发展后，到了 1987 年推出 X11 这个优秀的版本， 几经改良后，再推出了 X11R6 (X11 版本的第 6 次 release 的意思)这个版本。 目前我们看到的各大 Linux distributions 均是使用 X11R6 这个 X 版本哩！ (所以您才会常常看到您的 Linux 主机里面， 怎么会有这么多的 X11 与 X11R6 的目录啊！呵呵！)

X11 发展了一段时间后，由社群发起的一个称为 XFree86 的计划 (http://www.xfree86.org/) 持续在维护 X11R6 的功能性，包括对新硬件的支持以及更多新增的功能等等。 当初定名为 XFree86 其实是根据『 X + Free software + x86 硬件 』而来的呢。早期 Linux 所使用的 X Window 的主要核心都是由 XFree86 这个计划所提供的，因此，我们常常将 X server 与 XFree86 挂上等号的说～ 而 XFree86 针对 X11R6 也有持续在改良，也推出了 version 3 与 version 4 两个版本， 目前我们看到的新的 distribution 几乎

都是使用 version 4 版本， 不过在 2001 年以前的版本，则通常还是使用 XFree86 version 3 的版本呢！

除了 XFree86 这个计划之外，在 2004 年成立了另一个维护 X 系统架构的计划，那就是 Xorg (http://www.x.org/) 这个计划啦！Xorg 是由多个组织所共同发起的，主要的目的是希望可以持续维护 X11R6 ，使可以让 X Window System 更有效率的应用在工程上面。同时， Xorg 也是自由软件喔！ ^_^

Tips:
基本上，目前我们称为 X Window system 的，应该是 X11R6 这个版本的 X 系统。 而针对这个系统来发展的除了 XFree86 计划之外，在 2004 年产生的 Xorg 这个计划也对 X11R6 来进行维护与发展。会产生两个计划来维护 X11R6 的可能原因应该是目的的不同， Xorg 官方网站上面明白的宣示，该计划主要是希望可以让工程应用更加的有效率～ 而我们的 Fedora Core IV 使用的是 Xorg 计划所维护的 X11R6 版本喔！^_^



此外，XFree86 与 Xorg 针对他们自己发展的 X11R6 都有版本的区分，版本的编号是不一样的。举例来说，XFree86 分为 3.xx 与 4.xx 版本， 而 Xorg 则为 6.8.xx 等等。

X Server / X Client / Window manager 的关系

X Window System 为什么这么优秀呢？因为他有相当优良的 X Server/X Client 设计系统。 什么是 X Server 与 X Client 呢？X 在设计的时候， 就希望可以达到多人联机进入主机利用图形界面的功能，于是他便发展出这样的一个主从架构。这个主从架构可以让使用者在任何一部计算机以网络的方式联机到主机来操作图形界面的功能， 是一个相当棒的设计呢～不过 X Server 与 X Client 的意义则与网络上的 Server/Client 意义不同喔～底下我们就来谈一谈 X Server/X Client 的功能。

- X Server：主要负责的是屏幕画面的绘制与显示。

X Server 的主要功能 (不论是 Xorg 或是 XFree86 都是一个 X server 喔！) 就是在管理 X Server 所在主机上面关于显示的硬件配备啦～ 例如显示卡、屏幕分辨率、键盘形式、鼠标形式等等。如果以 Linux 上安装 X server 为例， 您会发现～咦！显示卡、屏幕以及键盘鼠标的设定，不是在开机的时候， Linux 系统以 /etc/sysconfig 目录下的 keyboard/mouse 等设定档就设好了吗？ 呵呵～这是因为 X Window 在 Linux 里面仅能算是『一套很棒的软件』， 所以 X Window 有自己的设定档，您必须要针对他的设定档设定妥当才行。

也就是说， Linux 的设定与 X Server 的设定不一定要相同的！ 因此，你在 Linux 的 run level 3 想要玩图形接口时，就得要加载 X Window 需要的驱动程序才行～总之， X Server 的主要功能就是在管理『主机』上面的显示硬件与驱动程序。

您会发现鸟哥一直强调一件事情，那就是『 X Server 所在的主机』，这是怎么回事啊？！ 刚刚我们不是提到 X Window System 最早是希望可以达到多人联机的目的吗？！ 就是很多人都可以使用 X 来联机进入主机的意思，但是每部联机进入 Linux 主机的客户端计算机的硬件并不一样啊～这个时候当然就无法使用 Linux 上面的硬件来显示数据啰～您说对吧？！否则您在客户端利用 1MB 显示卡内存的系统， 可能显示出 Linux 主机的 1024x768x24bit 色彩与分辨率吗？！当然不行～对吧！？ 也就是说：『每部想要显示 X 的主机都需要有 X Server』啦～ 所以，您的 Window PC 当然也需要 X Server 来管理显示接口，这样才

能够与 Linux 主机进行图形接口的沟通啊！更多的关于客户端联机到 Linux 主机端的方法，请参考 鸟哥的 Linux 私房菜--服务器篇：远程联机服务器 的介绍啰～

X Server 还有一个重要的工作，那就是将来自输入装置(如键盘、鼠标等)的动作告知 X Client， 您晓得，X Server 既然是管理这些周边硬件，所以，周边硬件的动作当然是由 X Server 来管理的， 但是 X Server 本身并不知道接口设备这些动作会造成什么显示上的效果， 因此 X Server 会将接口设备的这些动作行为告知 X Client ，让 X Client 去伤脑筋～

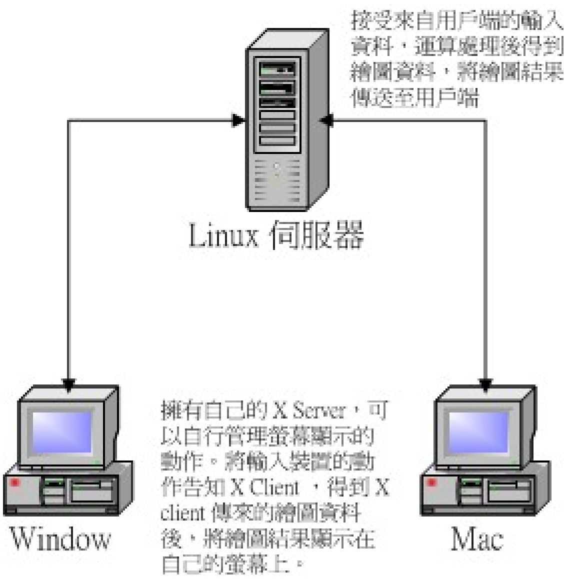
- X Client：主要负责的是『事件』的处理。

前面提到的 X Server 主要是管理显示接口与在屏幕上绘图，同时将输入装置的行为告知 X Client， 此时 X Client 就会依据这个输入装置的行为来开始处理，最后 X Client 会得到『 嗯！这个输入装置的行为会产生某个图示』，然后将这个图标的显示数据回传给 X Server ， X server 再根据 X Client 传来的绘图资料将他描图在自己的屏幕上，来得到显示的结果。

也就是说， X Client 最重要的工作就是处理来自 X Server 的动作，将该动作处理成为绘图数据， 再将这些绘图数据传回给 X Server 啰～

举个例子来说，当我们在 X Window 的画面中，将鼠标向右移动，那他是怎么告知 X Server 与 X Client 的呢？ 首先， X server 会侦测到鼠标的移动，但是他不知道应该怎么绘图啊！此时，他将鼠标的这个动作告知 X Client， X Client 就会去运算，结果得到，嘿嘿！其实要将鼠标指针向右移动几个位素，然后将这个结果告知 X server ， 接下来，您就会看到 X Server 将鼠标指针向右移动啰～

这样做有什么好处啊？！最大的好处是， X Client 不需要知道 X Server 的硬件配备与操作系统！ 因为 X Client 单纯就是在处理绘图的数据而已，本身是不绘图的。所以，在客户端的 X Server 用的是什么硬件？ 用的是哪套操作系统？主机端的 X Client 根本不需要知道～相当的先进与优秀～对吧！ ^_^ 整个运作流程可以参考下图：客户端用的是什么操作系统在 Linux 主机端是不在乎的！



图一、X Window 与 X Client 的沟通示意图

刚刚好～在 Linux 上的 X Window System 中，X Server 与 X Client 在同一部 Linux 上面～ 真是刚好

啊～呵呵～好了，接下来再来讨论一下，那么那个 Window Manager 又是啥咚咚？ Window Manager 可以说是一个相当特殊的 X client ，他可以提供更多的功能～ 包括有：

- 提供许多的控制元素，包括工作列、背景桌面的设定等等；
- 管理虚拟桌面 (virtual desktop)；以及
- 提供窗口控制参数，这包括窗口的大小、窗口的重迭显示、窗口的移动、窗口的最小化等等。

刚刚前面提到，X Client 的主要工作是将来自 X Server 的数据处理成为绘图数据，再回传给 X server ，所以， X client 本身是不知道他在 X Server 当中的位置、大小以及其它相关信息的。 这些窗口位置大小与重迭显示的功能，还有每个窗口上头的标题等等，其实就是由 Window manager 所提供的啰～

FC4 利用 Xorg 这个计划提供 X server 的核心，同时 Xorg 也提供了一个简单的 Window manager ， 那就是 twm 啰。不过， twm 的功能虽然已经具备了 window manager 最阳春的能力，不过，就是太阳春～ 所以后来预设的 window manager 大致上都以 KDE 或者是 GNOME 这两个计划提供的 window manager 为主啰～这两个计划的目的都是要让使用者可以在 Linux 底下使用完整的桌面计算机能力， 因此这两个计划在 Window manager 底下增加了很多的 X Client 软件，而且也加入了很多办公应用软件， 来让大众接受 Linux 这个好东西啊！

那么您知道 X Server / X client / window manager 的关系了吗？！我们举 KDE 为例好了， 由于我们要在本机端启动 X Window system ，因此，在我们的 FC4 主机上面必须要有 Xorg 的 X server 核心， 这样才能够提供屏幕的绘制啊～然后为了让窗口管理更方便，于是就加装了 KDE 这个计划的 window manager ， 然后为了让自己的使用更方便，于是就在 KDE 上面加上更多的应用软件，包括输入法等等的，最后就建构出我们的 X Window System 啰～ ^_^

X Window 启用的流程

接下来，我们来谈一谈，那么您的 Linux 上头的 X Window System 整个启动流程是如何呢？ 假设您是以 run level 3 登入的好了，那么您要如何进入 X Window System 呢？ 很简单啊！输入 startx 就可以啦～ 那您知道 startx 执行后，您的 Window manager 是 KDE 还是 GNOME ？ 还是其它的 window manager 吗？！还有，您的版面风格配置又是如何呢？这些数据放在哪里啊？！ 底下我们就来谈一谈啰～

- 开始的一个侦测界面， startx ：

我们知道了 X Window system 其实就是 X Server + X client 嘛！然后， 为了让窗口管理更方便，于是在 X Client 加载 window manager 就是了。 不过，为了要加载这些数据，就必须要读取设定档对吧！我们知道 Linux 底下每个人都可以设定好属于自己的环境， X Window 也一样，您可以有自己专属的 X 画面。 但是，如果您是首次登入 X 呢？也就是说，您自己还没有建立自己的专属 X 画面时， 系统又是从哪里给你这个 X 预设画面呢？

事实上，当您输入 startx 时，这个 startx 的作用就是在帮您设定好上头提到的这些动作啰！ startx 其实只是一个 shell script ，他是一个比较亲和的程序，会主动的帮忙使用者建立起他们的 X 所需要引用的设定档而已。您可以自行研究一下 startx 这个 script 的内容， 鸟哥在这里仅就 startx 的作用作个介绍。

startx 在执行的时候，他会主动的去寻找使用者家目录底下的 .xinitrc 及 .xserverrc 这两个档案，这两个档案分别是：

- ~/.xinitrc 是 X Client 的设定数据文件;
- ~/.xserverrc 则是 X Server 的设定数据文件。

但是您不见得会有这两个档案,如果没有的话,那么 startx 就会主动的以系统预设的设定文件来启用,这两个设定档分别在 /etc/X11/xinit/xinitrc 与 /etc/X11/xinit/xserverrc。不论是您自己的 ~/.xinitrc 或者是系统的 xinitrc ,反正最后就是会有一个 X Client 与一个 X Server 的设定档会被取用,而您家目录下的 .xinitrc 与 .xserverrc 是具有优先权就是了。 在取得这两个东西之后,接下来就是以 xinit 这个程序来执行 X Client 与 X server 的启动了!

除此之外, startx 后面还可以接参数喔!这些参数可以取代 .xinitrc 与 .xserverrc 的设定。 举例来说,您想要让您的 X 色彩深度是 16 bit (色彩深度就是所使用的色彩啦!),那么就可以:

```
[root@linux ~]# startx [X client option] -- [X server option]
[root@linux ~]# startx  --  -depth 16
```

startx 后面接的参数以两个减号『--』隔开,前面的是 X Client 的设定,后面的是 X Server 的设定。因为色彩深度是与 X Server 有关的,所以参数当然是写在 -- 后面啰,于是就成了上面的模样! 由于 startx 后面加的参数可以取代默认值,因此您就可以使用 16 位色彩度进入 X 啰。

- 开始启动 X 的 xinit :

事实上,实际在启动 X 的,就是 xinit 这支程序啦~他的语法是:

```
[root@linux ~]# xinit [client option] -- [server or display option]
```

那个 client option 与 server option 如何下达呢?其实那两个咚咚就是由刚刚 startx 去找出来的啦!在我们透过 startx 找到适当的 xinitrc 与 xserverrc 后,就交给 xinit 来执行。在预设的情况下 (使用者尚未有 ~/.xinitrc 等档案时),您输入 startx , 就等于进行 xinit /etc/X11/xinit/xinitrc -- /etc/X11/xinit/xserverrc 这个指令一般!这样瞭了吗?

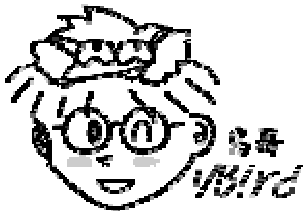
所以呢,重点当然就是 /etc/X11/xinit/ 目录下的 xinitrc 与 xserverrc 这两个档案的内容是啥啰~底下我们就分别来谈一谈这两个档案的主要内容与启动的方式~

- 启动 X Client 的档案: xinitrc :

假设您的家目录并没有 ~/.xinitrc ,则此时 X Client 会以 /etc/X11/xinit/xinitrc 来作为启动 X Client 的预设 script 。xinitrc 这个档案会将很多其它的档案参数引进来, 包括 /etc/X11/xinit/xinitrc-common 与 /etc/X11/xinit/Xclients 还有 /etc/sysconfig/desktop 。您可以参考 xinitrc 后去搜寻各个档案来了解彼此的关系。

重点是, xinitrc 会依据上述这些档案的判断 (要注意,每种 distributions 他们的 设定档案放置的地点都不太一样~) ,来搜寻出要启动的 window manager 是哪一个? 举例来说,在 FC4 底下, xinitrc 这个档案会经由分析出 /etc/sysconfig/desktop 的设定, 来开始执行 startkde 或者是 gnome-session 这两个 window manager 其中之一。 意思是说,如果您在 /etc/sysconfig/desktop 设定是 KDE 的话,那么预设就会以 KDE 来启动您的 X Window 啰。 等一下我们会以 KDE 的流程来介绍整个 X Client 如何启动的过程,这里目前仅指出到这里。 而在 X Client 执行完毕后,接下来,当然就是 X Server 的进程了!

Tips:
不论怎么说，鸟哥还是希望大家可以透过解析 startx 这个 script 的内容去找到每个档案， 再根据分析每个档案来找到您 distributions 上面的 X 相关档案～ 毕竟每个版本的 Linux 还是有所差异的～



在上面的步骤中，我们会看到 xinitrc 引入了 /etc/sysconfig/desktop 的设定， 已取得系统预设的 window manager 之后，接下来则是正确的分析该 window manager 是否存在， 若存在则尝试启动，若不存在则以其它存在的 window manager 来尝试启动。 因此，即使您 /etc/sysconfig/desktop 设定错误，系统还是会以预设的可能存在的 window manager 来尝试启动的。

如果是以 KDE 来启动的话，系统就会主动去搜寻 startkde 这个执行档。其实 startkde 也只是一个 script， 他包含了很多 KDE 需要的设定数据，详细的 KDE 相关请参考 KDE 的官方网站啰～

- 启动 X Server 的档案： xserverrc ：

如果您去查阅 /etc/X11/xinit/ 目录的话，会发现，根本就没有 xserverrc 这个档案啊！ 那我家目录也没有 .xserverrc ，这个时候系统会怎么做呢？其实单纯只是执行 xinit 的时候， 系统的预设 X Client 与 X Server 的内容是这样的：

```
xinit xterm  -geometry  +1+1  -n  login  -display  :0 --  X  :0
```

那个 xterm 是 X 窗口底下的虚拟终端机，后面会接一个『 -display :0 』 表示这个虚拟终端机是启动在第 :0 号显示接口的意思。而我们启动的 X server 程序就是 X 啦！ 其实 X 就是 Xorg 或 XFree86 的连结档，亦即是 X Server 的主程序啰！ 所以我们启动 X 还挺简单的～直接执行 X 而已。如果单纯以上面的内容来启动您的 X window 时， 您就会发现 tty7 有画面了！只是.....很丑～因为我们还没有启动 window manager 啊！

在启动 X Server 时，会去读取 X Server 的设定档，在 Xorg 使用的是 /etc/X11/xorg.conf 这个， 至于 XFree86 则是使用 /etc/X11/XF86Config (注意大小写) 这个设定档。针对这个设定档的内容， 我们会在下个小节介绍。反正 X Server 读取设定档如果一切 OK 就会在 tty7 顺利启动啰～ 而刚刚执行的 X Client 就会将绘图数据传送给 X Server 呢！最终您就能看到漂亮的 X 啰～

不过要注意的是，如果您的 xinitrc 设定档里面有启动的 x client 很多的时候，千万注意将除了最后一个 window manager 或 X Client 之外，都放到背景里面去执行啊！举例来说，像底下这样：

```
xclock -geometry 100x100-5+5 &
xterm -geometry 80x50-50+150 &
exec /usr/X11R6/bin/twm
```

意思就是说，我启动了 X ，并且同时启动 xclock / xterm / twm 这三个 X clients 喔！ 如此一来，您登入 X 就有这三个咚咚可以使用了！如果忘记加上 & 的符号，那就..... 会让系统等待啊，而无法一次就登入 X 呢！

最后我们知道，透过 startx 可以取得 X Client 与 X Server 的相关设定资料，亦即 xinitrc 与 xserverrc 这两个档案。 这两个档案可以让 xinit 这支程序来启动我们的 X Window ，而透过 xinitrc 可以设定需要启动的 window manager 是哪一个，至于透过 xserverrc 则能了解 X Server 使用的是那个主程序。通通启用后，就可以得到我们的 X 啰。 更多的 xinit 用法可以参考 man xinit ，而 X 的用法则 man X 啰～

另外，其实 X Server 是会启动至少一个 port 来监听 X client 的要求的，那就是预设的 port 6000 啰。不过，我们的 X 其实是很有弹性的，可以拥有多个 port 来监听不同 X Client 的需求， 这也是未来我们谈到服务器架设时 VNC (Virtual Network Computing) 服务器的特色。 但是在 X Window System 的环境下，我们称 port 6000 为第 0 个显示接口，亦即为 hostname:0 ， 那个 hostname 通常可以不写，所以就成了 :0 即可。

那么启动的 X 画面是放在哪一个终端机 (tty) 呢？在预设的情况下，第一个启动的 X (不论是启动在第几个 port number) 是在 tty7 ，亦即按下 [ctrl]+[Alt]+[F7] 那个画面。 而起动的第二个 X (注意到了吧！可以有多个 X 同时启动在您的系统上呢) 则预设在 tty8 亦即 [ctrl]+[Alt]+[F8] 那个画面呢！很神奇吧！ ^_^

因为主机上的 X 有多个，因此，当我们在启动 X Server / Client 时， 应该都要注明该 X Server / Client 主要是提供或接受来自哪个 display 的 port number 才行。 如果您的 X 启动在 :1 时，那就是 port 6001 啦!

好了，我们可以来针对 X Server 与 X client 的架构来做个简单的测试喔！底下这些动作您必须先以 run level 3 登入，并且确定主机上面已经安装了 X Window System ， 并且您必须要在主机前面，不可以 ssh 之类的联机程序进入做底下的动作啊！ 另外，如果是 FC4 的使用者，记得将 xfs 服务启动喔！

```
1. 先来启动第一个 X 在 :0 画面中：
[root@linux ~]# X :0 &
# X 是大写，那个 :0 是写在一起的，至于 & 则是放到背景去执行。
# 此时可以使用 netstat -tulnp 查看看有没有那个 port 6000 出现啊！
# 另外，此时系统会主动的跳到第一个图形接口终端机，亦即 tty7 上喔！
# 所以如果一切顺利的话，您应该可以看到一个 X 的鼠标光标可以让您移动了。
# 该画面就是 X Server 启动的画面啰！丑丑的，而且没有什么 client 可以用啊！
# 接下来，请按下 [ctrl]+[alt]+[F1] 回到刚刚下达指令的终端机：

2. 输入数个可以在 X 当中执行的虚拟终端机
[root@linux ~]# xterm -display :0  &
# 那个 xterm 是必须要在 X 底下才能够执行的终端机接口。
# 加入的参数 -display 则是指出这个 xterm 要在那个 display 使用的。
# 此时请按下 [ctrl]+[alt]+[F7] 去到 X 画面中，您会发现多了一个终端机啰～
# 不过，可惜的是，您无法看到终端机的标题、也无法移动终端机，
# 当然也无法调整终端机的大小啊！我们回到刚刚的 tty1 然后：
[root@linux ~]# xterm -display :0  &
# 又多一个终端机，去到 tty7 查阅一下。唉～没有多出一个终端机啊？
# 这是因为两个终端机重迭了～我们又无法移动终端机，所以只看到一个。

3. 输入可以管理的 window manager
[root@linux ~]# twm -display :0  &
# 回到 tty1 后，用最简单的 twm 这个窗口管理员来管理我们的 X 吧！
# 输入之后，去到 tty7 看看，用鼠标移动一下终端机看看？可以移动了吧？
```

```
# 也可以缩小放大窗口啰～同时也出现了标题提示啰～也看到两个终端机啦！
# 现在终于知道窗口管理员的重要性了吧？ ^_^

4. 增加另一个 X 在系统中
[root@linux ~]# X :1 &
# 如果您又重复执行一次 X 的话，那么这次的 X 图示就会出现在 tty8 底下，
# 那再输入一次呢？亦即 X :2 & 呢？很简单啊！ tty9 也有 X 出现啰～厉害吧！

5. 将所有刚刚建立的 X 相关工作全部杀掉！
[root@linux ~]# kill %6
[root@linux ~]# kill %5
[root@linux ~]# kill %4
[root@linux ~]# kill %3
[root@linux ~]# kill %2
[root@linux ~]# kill %1
```

很有趣的一个小实验吧～透过这个实验，您应该会对 X 使用的 port ， 与 Window manager 及 tty7 以后的终端接口使用方式有比较清楚的了解～加油！

与 X Server 有关的设定档

从前面的说明来看，我们知道一个 X 能不能启用，其实与 X Server 有很大的关系的。因为 X Server 负责的是整个画面的描绘， 所以没有成功启动 X Server 的话，即使有启动 X Client 也无法将图样显示出来啊。所以，底下我们就针对 X Server 的设定档来做个简单的说明，好让大家可以成功的启动 X Window System 啊。

基本上， X Server 管理的是显示卡、屏幕分辨率、鼠标按键对应等等，尤其是显示卡芯片的认识，真是重要啊。 此外，还有显示的字体也是 X Server 管理的一环。基本上， X 的设定档都是预设放置在 /etc/X11 目录下， 而相关的显示模块或上面提到的总总模块，则主要放置在 /usr/X11R6 底下。 比较重要的是字型文件与芯片组，她们主要放置在：

- 字型：/usr/X11R6/lib/X11/fonts
- 显示芯片：/usr/X11R6/lib/modules/drivers

在 FC4 底下，我们可以透过 chkfontpath 这个指令来取得目前系统有的字型档案目录。 这些都要透过一个统一的设定档来规范，那就是 X server 的设定档啦。不过 XFree86 与 Xorg 这两个计划的设定档档名不同， 虽然内容是差不多，不过，还是要来分辨一下才行。

手动修改 xorg.conf 或 XF86Config

其实想要知道你的系统里面 X Server 到底来自哪个计划，最简单的方法就是利用 X 这个指令来取得版本啦。 您必须以 root 的身分执行下列指令：

```
[root@linux ~]# X -version
```

```
X Window System Version 6.8.2
Release Date: 9 February 2005
X Protocol Version 11, Revision 0, Release 6.8.2
Build Operating System: Linux 2.6.9-1.906_ELsmp i686 [ELF]
Current Operating System: Linux linux.dmtsai.tw 2.6.12-1.1387_FC4
Build Date: 02 June 2005
Build Host: tweety.build.redhat.com

        Before reporting problems, check http://wiki.X.Org
        to make sure that you have the latest version.
```

有看到 http://wiki.X.Org 吧?那就是 Xorg 计划的网站啰。此时我们知道设定档就是 /etc/X11/xorg.conf 。 如果您的 X 是 XFree86 的话，那么设定档就会是在 /etc/X11/XF86Config (注意大小写) 里头喔! 我们的 FC4 使用的是 Xorg 计划所提供的 X11，所以我们的设定档就会在 /etc/X11/xorg.conf 。 接下来，我们要做的就只是修改 xorg.conf 这个档案而已。之前各大 distribution 提供的例如 xf86config, Xconfigurator 之类的软件也都是在修改这个设定档而已。所以，我们就直接手动来查阅这个档案吧。

Tips:
其实 xorg.conf 与 XF86Config 这两个设定档的内容几乎一模一样，所以，即使您的 X 是 XFree86 的话，同样可以适用底下的说明喔。



注意一下，在修改这个档案之前，务必将这个档案给她备份下来，免的改错了甚么东西，导致连 X server 都无法启动的问题啊。这个档案的内容是分成数个段落的，每个段落以 Section 开始，以 EndSection 结束， 里面含有该 Section (段落) 的相关设定值，例如:

```
Section “section name”
……..  <== 与这个 section name 有关的设定项目
……..
EndSection
```

至于常见的 section name 主要有:

1. Module: 加载到 X Server 当中的须要模块;
2. InputDevice: 包括输入的 1. 键盘的格式 2. 鼠标的格式;
3. Files:设定字型所在的目录位置等;
4. Monitor:屏幕的格式，主要是设定水平、垂直的更新频率;
5. Device:这个重要，就是显示卡芯片组的相关设定了;
6. Screen:这个是在屏幕上显示的相关分辨率与色彩度的设定项目;
7. ServerLayout:上述的每个项目都可以重复设定，这里则是此一 X server 要取用的那个项目值的设定啰。

好了，直接来看看这个档案的内容吧!鸟哥我的显示卡是 NVidia 的早期的 GeForce2 MX 的卡，那我的 xorg.conf 内容是这样的:

```
[root@linux ~]# cd /etc/X11
```

```
[root@linux X11]# cp -a xorg.conf xorg.conf.back  <== 有备份有保佑
[root@linux X11]# vi xorg.conf
Section "Module"
        Load  "dbe"
        Load  "extmod"
        Load  "fbdevhw"
        Load  "glx"
        Load  "record"
        Load  "freetype"
        Load  "type1"
        Load  "dri"
EndSection
# 上面这些模块是 X Server 启动时，希望能够额外获得的相关支持的模块。
# 关于更多模块可以搜寻一下 /usr/X11R6/lib/modules/extensions/ 这个目录，
# 对喜欢以 X 窗口登入远程主机的朋友来说， FC4 提供的这个 vnc.so 模块可是
# 相当的有趣喔!请参考 鸟哥的 Linux 私房菜服务器篇--远程登入服务器的介绍呢

Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "kbd"
        Option      "XkbModel" "pc105"
        Option      "XkbLayout" "us"
EndSection
# 这个玩意儿是键盘的对应设定数据，重点在于 XkbLayout 那一项，
# 如果没有问题的话，我们台湾地区应该都是使用美式键盘对应按钮的，
# 但是 SuSE 却是使用德国键盘对应按钮，这里老是发生错误，呵呵，所以，
# 如果你的键盘老是按出不对的字符，修改这里成为 us 吧!
# 特别注意到 Identifier 那一项，那个是在说明，我这个键盘的设定文件，
# 被定义为名称是 Keyboard0 的意思，这个名称最后会被用于 ServerLayout 中

Section "InputDevice"
        Identifier  "Mouse0"
        Driver      "mouse"
        Option      "Protocol" "IMPS/2"
        Option      "Device" "/dev/input/mice"
        Option      "ZAxisMapping" "4 5"
        Option      "Emulate3Buttons" "yes"
EndSection
# 这个则主要在定义鼠标。鸟哥用 ps2 鼠标，所以这里是这样的模样。
# 如果您使用的是 usb 鼠标，这里可能就不会是跟我一样的情况了。
# 比较有趣的是那个 Emulate3Buttons ，意思是模拟三键，
# 如果您的鼠标本身就有三按键，这里就不要设定啊
```

```
Section "Files"
        RgbPath      "/usr/X11R6/lib/X11/rgb"
        FontPath     "unix/:7100"
EndSection
# 我们的 X Server 很重要的一点就是必须要有字型，
# 这个 Files 的项目就是在设定字型的地方。当然啦，您的主机本来就必须
# 要有字型文件才行。一般字型文件就是放置在 /usr/X11R6/lib/X11/fonts。
# 那个 Rgb 是与色彩有关的项目。
# 基本上，如果我们要填写 100dpi 的字型，可以这样做:
# FontPath     "/usr/X11R6/lib/X11/fonts/100dpi/"
# 将所有需要的字型都重复以上面这一行填写即可。但是， FC4 有更好的方法，
# 那就是利用 X Font Server, xfs 这个 daemon 来统一管理，因此，
# 上面的设定值就会只剩下 "unit/:7100" 而已，那说明的是，我们的 xfs
# 服务是启动在 unix socket 7100 那个数据链路上面。
# 也因为如此，所以我们在启动 X server 之前，务必要先启动 xfs 才行啊，
# 否则 X 会告知我们说，找不到 font 呢。相关的信息我们在下一继续说明。

Section "Monitor"
        Identifier   "Monitor0"
        VendorName   "Monitor Vendor"
        ModelName    "ADI GD910T"
        DisplaySize  370        280
        HorizSync    30.0 - 80.0
        VertRefresh  50.0 - 100.0
        Option      "dpms"
EndSection
# 屏幕的设定仅有一个地方要注意，那就是垂直与水平的更新频率。
# 在上面的 HorizSync 与 VerRefresh 的设定上，要注意，不要设定太高，
# 以鸟哥的设定为例，我并没有设定很高喔，这个玩意儿与更新频率有关。
# 设定太高的话，据说会让 monitor 烧毁呢，要很注意啊。

Section "Device"
        Identifier  "Videocard0"
        Driver      "nv"
        VendorName  "Videocard vendor"
        BoardName   "NVIDIA GeForce 4 (generic)"
EndSection
# 这地方重要了，这就是显示卡的芯片模块加载的设定区域。
# 因为鸟哥的显示卡是 NVidia 的，所以就使用 Xorg 计划提供的 nv 模块，
# 那个模块就是给 NVidia 用的啦。更多的显示芯片模块可以参考底下这个目录:
# /usr/X11R6/lib/modules/drivers/

Section "Screen"
```

```
        Identifier "Screen0"
        Device     "Videocard0"
        Monitor    "Monitor0"
        DefaultDepth     16
        SubSection "Display"
                Viewport   0 0
                Depth     16
                Modes    "1024x768" "800x600" "640x480"
        EndSubSection
        SubSection "Display"
                Viewport   0 0
                Depth     24
                Modes    "1024x768" "800x600"
        EndSubSection
EndSection
# 那么 screen 是甚么东西啊? 其实就是与屏幕有关的
# 分辨率啦、色彩深度啦等等的，还有一个预设色彩深度 (DefaultDepth)。
# 您会发现，在 screen 里面还有一个 subsection "display" 对吧?
# 那个是段落内的段落，也就是说，screen 里面还可以设定多个显示项目，
# 以上面的段落来说，我的 screen (屏幕) 还可以针对 16bit 与 24bit (Depth)
# 色彩度来设定分辨率 (Modes) 呢。而预设的情况就是 16 bit 色度
# (看的是 DefaultDepth 那个项目啊!)
# 以鸟哥来说，我本人比较喜欢 16bit 色彩深度与 1024x768 的分辨率，因此，
# 鸟哥常常在这里仅设定一个 subsection "display" 而已，内容有点像:
# SubSection "Display"
#          Viewport   0 0
#          Depth                    16
#          Modes                    "1024x768"
# EndSubSection

Section "ServerLayout"
        Identifier      "Default Layout"
        Screen      0   "Screen0" 0 0
        InputDevice     "Mouse0" "CorePointer"
        InputDevice     "Keyboard0" "CoreKeyboard"
EndSection
# 我们上面设定了这么多的项目之后，最后整个 X Server 要用的项目，
# 就通通一骨脑的给他写入这里就是了，包括键盘、鼠标以及屏幕啊。
```

上面设定完毕之后，就等于将整个 X Server 设定妥当了，很简单吧。 如果您想要更新其它的例如显示芯片的模块的话，就得要去硬件开发商的网站下载原始档来编译才行。 设定完毕之后，您就可以启动 X Server 是看看啰。基本上，如果您的 Files 那个项目用的是直接写入字型的路径， 那就不需要启动 XFS (X Font Server)，如果是使用 font server 时，就要先启动 xfs :

```
[root@linux ~]# /etc/init.d/xfs start
```

```
[root@linux ~]# startx
```

当然，您也可以利用 init 5 这个指令直接切换到图形接口的登入来试看看啰。

X Font Server (XFS)

我们刚刚上面提到了要启动 X Server 时，根据的设定档是 /etc/X11/xorg.conf ， 但是在这个设定文件里面关于字型的设定可以使用 X Font Server 来统一管理。那么 xfs 的设定档又在哪里呢?这就涉及了字型文件放置的地点了。我们知道目前的字型放置在哪里呢?就是 /usr/X11R6/lib/X11/fonts/ 这个目录下的其它目录。 而统一管理 xfs 的启动程序是在 /etc/init.d/xfs ，分析其中的 script 结果知道设定档在 /etc/X11/fs/config ，查阅档案内容发现:

```
[root@linux ~]# vi /etc/X11/fs/config
# allow a max of 10 clients to connect to this font server
client-limit = 10

# when a font server reaches its limit, start up a new one
clone-self = on

# where to look for fonts
catalogue = /usr/X11R6/lib/X11/fonts/misc:unscaled,
        /usr/X11R6/lib/X11/fonts/75dpi:unscaled,
        /usr/X11R6/lib/X11/fonts/100dpi:unscaled,
        /usr/X11R6/lib/X11/fonts/Type1,
        /usr/share/fonts/default/Type1,
        ,
        /usr/share/fonts/zh_CN/TrueType,
        /usr/share/fonts/zh_TW/TrueType
# in 12 points, decipoints
default-point-size = 120

# 100 x 100 and 75 x 75
default-resolutions = 75,75,100,100

# use lazy loading on 16 bit (usually Asian) fonts
deferglyphs = 16

# how to log errors
use-syslog = on

# don't listen to TCP ports by default for security reasons
no-listen = tcp
```

上面这个档案的设定重点在 catalogue 那个设定项目当中。您可以使用 chkfontpath 这个指令来列出目前支持的字型档案，也可以直接修改呢！而您如果还想要使用其它的特殊按键，则可以尝试使用 xmodmap 哩！

/etc/inittab

我们由前面知道 run level 5 可以主动的进入 X 窗口系统， 而修改 run level 的设定数据可以查询 /etc/inittab 内的这个参数:

```
[root@linux ~]# vi /etc/inittab
id:3:initdefault:
.....(略)......
x:5:once:/etc/X11/prefdm -nodaemon
```

看到了吗?那个 id 开头的项目就是开机启动时预设的 run level 设定值， 至于那个 x 开头的设定项目，则是在说明如果以 run level 5 启动时， 需要额外执行的程序，就是 /etc/X11/prefdm 这个程序啰。 简单分析一下 prefdm 档案，内容其实也只是要藉由 /etc/sysconfig/desktop 这个设定档来分析出预设执行的 Window manager 啰!

利用 Xorg / XFree86 来设定预设的设定档

我们在上面提到了关于手动修改 xorg.conf 这个 X Server 的设定档。但是， 如果我对于我的系统内的所有相关硬件都不知道的时候，也不晓得如何主动建立这个档案时， 有没有比较快速的方法可以来建立出这个设定档呢? 是有的，同样也是 Xorg 的功能之一， 藉由让 Xorg 以预设的所有模块去进行整个系统的探索 (probe) 后，先主动尝试建立一个阳春的 X Server 设定档，然后您再藉由测试与手动修改来修订这个设定档呢。如何执行呢? 您必须以 root 的身分执行才行:

```
[root@linux ~]# Xorg -configure
# 此时 X 会主动的以内建的模块进行系统硬件的探索，
# 并将最后的结果输出到 /root/xorg.conf.new 这个档案里面去。
```

整个过程很简单，就是利用 Xorg -configure 而已，如果是 XFree86 的话，同样可以使用 XFree86 -configure 来建立起自己的设定档。而以 Xorg 建立的设定档会是 /root/xorg.conf.new 这个档名，你可以在 run level 3 的环境下，输入:

```
[root@linux ~]# X -config /root/xorg.conf.new
```

来测试一下这个设定档是否能够启动呢? 如果输入上面指令后，可以在 tty7 看到画面的话， 表示这个设定档就应该可以启动 X Server 啰。果真如此的话，您就可以将这个档案覆盖成 /etc/X11/xorg.conf ，并且在自己手动手动微调里面的设定值吧。

更新显示卡驱动程序的范例： NVidia 驱动程序

在很多情况下，您的 X Server 可能需要针对显示卡芯片做个升级的。举例来说， 您购买的显示卡是最新的，但是 X Server 的预设模块并没有支持。又或者您所需要的某些显示芯片功能中， 预设的 X Server 的模块并无法提供。记得鸟哥在研究所所进行的研究中，有一项工作是需要在 Linux 服务器上面启用一个图形处理软件，该软件虽然是自由软件，不过，它需要 X Server 的 GLX 功能的支持， 我们实验室的 NVidia 是有支持该功能的，偏偏预设的 X server 的 nv 模块并不提供该功能， 所以鸟哥就得到 NVidia 官方网站下载最新的驱动程序啦......

在这里，鸟哥以时下最流行的显示芯片制造商之一的 NVidia 来约略说明一下升级芯片模块（或者说是驱动程序）的简单流程，希望对大家能有点帮助啊。

- 下载驱动程序

既然要更新驱动程序，当然要到硬件开发商的网站下载啰。 您可以到底下的网站去下载驱动程序。

http://www.nvidia.com/content/drivers/drivers.asp

在上面连结的框框中，由左至右依序选择{Graphics Driver} {Geforce and TNT2} {Linux IA32} 要注意的是，因为鸟哥的显示卡是 GeForce 系列的卡，而我的操作系统 FC4 与硬件搭配， 其实是 32 位，那个 IA32 亦即是 32 位的操作系统。至于 Linux AMD64 与 Linux IA64 则分别代表 AMD 64 位的硬件以及其它 64 位的操作系统。点选完毕后按下 Go 按键， 就会进入说明画面。里面就会有驱动程序的下载联结。

在该网页当中，还会有很多的信息，您应该要看一看的，里面会有教您如何如何正确的安装该软件， 以及发生错误时应该如呵解决的方案喔。

- 开始安装模块

在该模块下载之后，您会发现他并不是一个 RPM 档案。根据 NVidia 的说法， 因为并非所有的 distribution 都是使用 RPM 作为套件管理的，所以她们就取消了单纯的 RPM 方式， 改以一个步骤就完成的方式来给予这个安装档案。

其实整个安装很简单，直接执行该下载档案就好了！下载的文件名称应该是:
NVIDIA-Linux-x86-{version}-pkg1.run 他是一个含有原始码的 shell script ，既然是 Shell script ，直接安装它就好啦！

```
[root@linux ~]# sh NVIDIA-Linux-x86-1.0-7667-pkg1.run
# 过程会出现一个授权说明，请选择 Accept 项目即可，
# 而接下来会出现一个找不到相关核心接口的说明，他说要去 NVidia 下载适当的接口。
# 此时可以选择不要，那么程序就会开始自动编译驱动程序的主核心啰.
```

这个时候程序会将 Nvidia 的驱动程序放置在 /usr/X11R6/lib/modules/drivers/nvidia_drv.o 而且更新了 GLX 这个模块在 /usr/X11R6/lib/modules/extensions/libglx.so.1.0.7667 。 并且同时会安装 NVidia 的 nvidia-bug-report.sh, nvidia-installer, nvidia-settings 等工具程序。比较直得注意的是 nvidia-installer 这支程序，他还可以帮忙在线更新哩~

```
[root@linux ~]# nvidia-installer --update
# 这个动作在进行在线更新的作业
[root@linux ~]# nvidia-installer --uninstall
# 这个动作在解安装 NVidia 的模块
```

呵呵，真是方便，未来要升级到更新的显示卡驱动程序， 直接利用 nvidia-installer 的功能即可。更多详细的用法可以参考 nvidia-installer -h 的说明。

- 修改主要设定档内容

接下来就是要让 X Server 去直接读取刚刚安装的模块了，所以，修改一下 xorg.conf 啰！

```
[root@linux ~]# vi /etc/X11/xorg.conf
# 在 Section "Device" 的显示卡芯片段落处，修改 driver 啰。
#       Driver      "nv"
```

```
        Driver      "nvidia"
# 将原先的 nv 模块改成 nvidia 模块喔

# 在 Section "Module" 的地方，取消一些预设的模块，改以 NVidia 的预设功能。
        Load  "glx"
# 上面这一段如果不存在，就手动建立吧
#       Load  "dri"
#       Load  "GLCore"
# 如果有出现 Load dri 与 GLCore ，请将他批注掉。
```

到此为止，就已将您的系统安装上最新的 NVidia 的显示卡驱动程序啰。 好啰，那您就赶紧试看看新的显示卡芯片的功能吧。 而如果有什么疑问的话，查阅一下 /var/log/nvidia 开头的登录档看看吧！ ^_^

问题克服

- 以前可以进入 X Window System ，而我没有做过甚么额外的行为，不过，最近却无法进入 X Window 了，该怎么办?
  最可能发生的原因是因为您在进行某些动作时让 KDE 的预设档跑掉了， 如果没有甚么重要的设定参数，建议您可以将您家目录下的 ~/.kde 或者是 ~/.fonts ， 或者是 GNOME 在您家目录下的 ~/.gnome* 目录给他移除，让系统自动帮您建立一个全新的设定档试看看。
- 我老是没有办法启动 X Window System ，不知道原因出在哪里呢?
  出现这样的问题最麻烦，要一个一个去检查问题点。首先，如果您使用的是 Xorg 的话， 先查阅一下 /var/log/Xorg.0.log 这个登录文件的信息，他几乎会告知您所有 X Server 发生的错误，经由这个档案的告知，您应该可以克服大部分的 X Server 的困扰。接下来的 X Client 问题， 可以查阅 /var/log/kdm.log 等其它与 X Client 软件有关的登录文件。 而如果是无法了解 X Server 的问题，那么建议您使用 Xorg -configure 建立一个全新的设定档后， 再逐一修改比对您所设定的参数，以了解错误发生的原因啰。
- 我的 X Server 里头，使用 KDE 或 GNOME 都无法更新或调整屏幕分辨率及更新频率，该如何是好?
  发生这个问题的原因很多，不过，基本上都是与 X Server 有关，而不是 KDE 或 GNOME 的问题呢。 您应该要检查 xorg.conf 或 XF86Config 这两个档案，找到与 Screen 有关的 Section ，查询一下是否 display 那个 subsection 只有一个设定值?果真如此的话，当然您就无法调整分辨率了。 请手动增加其它分辨率与色彩度即可。另外，如果这部分没有问题，那就可能是出现在显示芯片模块的问题了。 或者需要到芯片开发商的网站上面下载最新的驱动程序来安装，应该可以解决您的问题。 当然，也可以到 /var/log/Xorg.0.log 或 /var/log/XFree86.0.log (看各 distributions 的设定) 查阅问题点。
- 如何进行 X Window System 的中文化?
  基本上，中文化应该都是在 KDE 或者是 GNOME 上面的显示软件才对。这里有一篇 FreeBSD 的文章： FreeBSD Chinese How-to (http://www.douzhe.com/docs/freebsd_howto/index.html) 虽然是 FreeBSD 的系统，不过，很有可观之处。

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 如何在 Linux 主机进入 X Window System ?

  如果是在 run level 3 ，可以使用 startx 进入，至于 run level 5 ，则直接进入 tty7 即可进入 X Window 系统画面。

- 利用 startx 可以在 run level 3 的环境下进入 X Window 系统。请问 startx 的主要功能？

  其实整个 X 系统的启动应该是由 xinit 这个指令所启发的。但 xinit 需要 X Client 与 X Server 的相关参数，以提供进入 X Window System 的软件与硬件管理，例如 xinit xinitrc -- xserverrc 等。 startx 即是在判断使用者是否有自己的 ~/.xinitrc 及 ~/.xserverrc 的 script， 若有则直接取用，若无则到 /etc/X11/xinit 底下取用。

- 如何知道您系统当中 X 系统的版本与计划？

  最简单可以利用 root 的身份下达 X -version 即可知道！

- 要了解为何 X 系统可以允许不同硬件、主机、操作系统之间的沟通，需要知道 X server / X client 的相关知识。 请问 X Server / X client / Window manager 的主要用途功能？

  X Server 主要负责屏幕的绘制，以及周边输入装置如鼠标、键盘等数据的收集，并回报给 X Client ； X Client 主要负责数据的运算，收到来自 X Server 的数据后，加以运算得到图形的数据，并回传给 X Server， 让 X server 自行绘制图形。至于 Window manager 是一个比较特殊的 X Client ，他可以管理更多控制元素， 最重要的地方还是在于窗口的大小、重迭、移动等等的功能。

- 如何重新启动 X

  最简单在 X Window System 下，直接按下 [alt]+[ctrl]+[backspace<--] 即可， 也可以 init 3 再 init 5，也可以关闭 X 后，再 startx 启动等等。

- 试说明 ~/.xinitrc 这个档案的用途？

  当我们要启动 X 时，必须要启动 X Client 软件端。这个 ~/.xinitrc 即是在客制化自己的 X Client ， 您可以在这个档案内输入您自己的 X Client 。若无此档案，则预设以 /etc/X11/xinit/xinitrc 替代。

- 我在 FC4 的系统中，预设使用 KDE 登入 X 。但我想要改以 GNOME 登入，该怎么办？

  最简单的作法，直接修改 /etc/sysconfig/desktop 内的设定值即可。 但如果你不是 root 无法修订该档案时， 亦可以在自己的家目录参考 /etc/X11/xinit/xinitrc 的内容自行制作 ~/.xinitrc 档案来修改！

- X Server 的 port 预设开放在？

  X port 预设开放在 port 6000 ，而且称此一显示为 :0

- Linux 主机是否可以有两个以上的 X

  是的！可以！第一个 X 通常在 tty7 ，第二个在 tty8 ，第三个在 tty9 ，依序类推。 第几个是以启动的顺序来定义，并非 :0 , :1 的意思～特别分清楚。

- X Server 的设定档若不是 xorg.conf 就是 XF86Config 。 在该档案中， Section Files 干嘛用的？

  相当重要！是设定显示字型用的。而字型一般放置目录在 /usr/X11R6/lib/X11/fonts/ 当中。

- 我发现我的 X 系统键盘所输入的字母老是打不出我所需要的单字，可能原因该如何修订？

  应该是键盘符号对应表跑掉了。可以修改 xorg.conf 或 XF86Config 档案内，关于 Keyboard 的 Option XkbLayout 项目，将他改为 us 即可！

- 当我的系统内有安装 GNOME 及 KDE 两个 X Widnow Manager ，我原本是以 KDE 为预设的 WM，若想改为 GNOME 时，应该如何修改 (假设在 FC4 的环境下)？

  每个 distributions 的修改方式都不太一样，以 FC4 为例，修改 /etc/sysconfig/desktop 内部，成为 GNOME 即可！而 SuSE 可以修改 /etc/sysconfig/windowmanager ！

参考数据

- 王垠先生对 X Window 的介绍 http://learn.tsinghua.edu.cn/homepage/2001315450/x.html
- X.org 官方网站 http://www.x.org/
  里面有很多不错的文件可以参考喔！
- XFree86 官方网站 http://www.xfree86.org/
- 一篇对 X Window system 颇多关联的文章 http://linux.xab.ac.cn/bbs/read.php?tid=352
- man Xorg
- man startx
- man xinit
- 关于 FC4 的图形接口登入时，产生的一些 tty 画面困扰：
  https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=161242
  简单的来说，就是 Intel 的显示卡与 G550 的显示卡硬件在 FC4 的编译过程中， 可能无法对这两种显示卡做比较好的处理，导致 tty1 ~ tty7 都没有东西跑出来。 可以将 FC3 的这个档案 /usr/X11R6/lib/modules/libvgahw.a 复制到 FC4 底下的相同档案去， 重新开机后，应该可以克服这个问题。该档案可以在底下取得：
  ftp://people.redhat.com/mharris/libvgahw.a

Linux 硬件侦测与维护

最近更新日期：2005/11/25

谈完了 Linux 的系统操作，再来得要了解一下若您的主机硬件更换，或者是想要进行主机的温度侦测时，应该要怎么作？在这个章节当中，我们会谈到如何以 Linux kernel 里面的 lm_sensor 功能， 以及利用 LVM 来制作一个大型的硬盘，让您的 filesystem 容量更具有弹性喔！ 当然啦，在 Linux 系统上面接上您的打印机，也是一个很好玩的工作喔！那就来读一读吧！

准备好你的硬件：

我们这个章节的主要目的在于更深入的了解我们 Linux 主机的硬件，并进一步的将这些硬件信息汇整，最好是还能够进行一些侦测与控管啦！这样才可以在最短的时间内了解我们 Linux 主机是否可能发生问题啊！ 这包括主机的电压值是否正确？主机与 CPU 温度是否过高等等。而为了这些目的，当然啦， 我们就得要先准备好自己的 Linux 主机的硬件啊，至少得要先了解你的 Linux 主机是啥硬件配备啊， 否则，假如你的硬件本身就是不支持温度侦测，那花再多时间在温度数值的取得，嘿嘿！是没有意义的啦！ 来给他进行下去啰！ ^_^

硬件信息的收集： lspci, iostat...

还记得我们在 开机与关机流程分析 当中提到的核心功能吧？ 我们的 Linux kernel 会在开机时，以核心内建的功能去侦测主机的种种硬件， 并尝试加载适当的驱动程序 (模块, modules) 来让硬件正确的启动与运作。 而核心所侦测到的各项硬件装置，后来就会被记录在 /proc 当中了。 包括 /proc/cpuinfo, /proc/partitions, /proc/interrupts 等等。 更多的 /proc 内容介绍，先回到 程序与资源管理 那一篇去瞧一瞧先！

Tips:
当然还是得要再提到，核心所侦测到的硬件可能并非完全正确喔！他仅是『使用最适当的模块来驱动这个硬件』而已，所以有时候难免会误判啦！ 所以啰，如果你对于系统的稳定性是斤斤计较的，那么或者重新编译一次你的核心， 应该也是个不错的主意。但是，一般来说，我们没有很建议你一定要重新编译核心就是了。 关于核心的编译，瞧一瞧下个章节吧！ ^_^



我们的 Linux 基本上有提供几个简单的指令来将核心所侦测到的硬件叫出来的～ 这包括我们之前提到的：

- hdparm：观察硬盘的种种信息；
- lspci：检查整个系统 PCI 接口的各项装置！很有用的指令；
- dmesg：观察核心运作过程当中所显示的各项讯息记录；
- iostat：检查整个 CPU 与接口设备的 Input/Output 状态。

无论如何，那个 lspci 真的是一个很不错用的指令，他可以直接将 /proc 底下的关于 PCI 接口的各项数据一口气的将他完整的呈现在你面前，可以让您很快速的了解到核心所侦测到的你的主机硬件呢！ 那么就赶紧来看一看你的主机硬件配备吧！ ^_^

基本上，想要知道您 Linux 主机的硬件配备，最好的方法还是直接拆开机壳去察看上面的信息， 如果环境因素导致您无法直接拆开主机的话，那么直接 lspci 是很棒的一的方法：

- lspci

```
[root@linux ~]# lspci [-vvn]
参数：
-v  ：观察更多的 PCI 装置的信息；
-vv ：比 -v 还要更详细的细部信息；
-n  ：直接观察 PCI 的 ID 而不是厂商名称
范例：

范例一：查阅您系统内的 PCI 装置：
[root@linux ~]# lspci
00:00.0 Host bridge: VIA Technologies, Inc. VT82C693A/694x [Apollo PRO133x] (rev c4)
00:01.0 PCI bridge: VIA Technologies, Inc. VT82C598/694x [Apollo MVP3/Pro133x AGP]
.....(中间省略).....
00:0c.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139C+ (rev 10)
01:00.0 VGA compatible controller: nVidia Corporation NV17 [GeForce4 MX 440] (rev a3)
# 不必加任何的参数，就能够显示出目前主机上面的各个 PCI 接口的装置呢！
# 从上面的数据可以看出我的显示卡是 Nvidia 的，主机芯片则是 VIA 的，
# 网络卡则是 Realtek 的 8139 (亦即是螃蟹卡啊！)。
```

基本上，我们可以由 lspci 立刻得到主机上面的各项设备，如果还想要知道他所占用的 IRQ 与 I/O port 时，可以加上『 lspci -vv 』来查阅啊！简单得不得了吧！ 另外，您会发现上表当中的每一行最前面都有一组怪怪的数字，那是什么？ 其实那就是我们的硬件侦测的数据啦！您可以对照着底下这个档案来查阅：

- /usr/share/hwdata/pci.ids

那个就是 PCI 的标准 ID 与厂牌名称的对应表啦！ 此外，刚刚我们使用 lspci 时，其实所有的数据都是由 /proc/bus/pci/ 目录下的数据所取出的呢！了解了吧！ ^_^

- iostat

刚刚那个 lspci 找到的是目前主机上面的硬件配备，那么整部机器的储存设备， 主要是硬盘对吧！请问，您硬盘由开机到现在，已经存、取多少数据呢？ 这个时候就得要 iostat 这个指令的帮忙了！

```
[root@linux ~]# iostat [-c|-d] [-k] [-t] [间隔秒数] [侦测次数]
参数：
-c  ：仅显示 CPU 的状态；
-d  ：仅显示储存设备的状态，不可与 -c 一起用；
-k  ：预设显示的是 block ，这里可以改成 K bytes 的大小来显示；
-t  ：显示日期出来；
范例：

范例一：显示一下目前整个系统的 CPU 与储存设备的状态
[root@linux ~]# iostat
Linux 2.6.12-1.1456_FC4 (localhost.localdomain)    10/27/05

avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.34    0.01    0.34    0.20   99.11

Device:            tps   kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
hda               0.55         2.64         3.28    6631507    8230880
hdb               0.21         4.52         9.10   11342022   22855752
# 瞧！上面数据总共分为上下两部分，上半部显示的是 CPU 的当下信息；
# 下面数据则是显示储存装置 /dev/hda, /dev/hdb 的相关数据，他的数据意义：
# tps       ：平均每秒钟的传送次数！与数据传输『次数』有关，非容量！
# kB_read/s ：开机到现在平均的读取单位；
# kB_wrtn/s ：开机到现在平均的写入单位；
# kB_read   ：开机到现在，总共读出来的档案单位；
# kB_wrtn   ：开机到现在，总共写入的档案单位；

范例二：每两秒钟侦测一次，并且共侦测三次储存装置
[root@linux ~]# iostat -d 2 3
Linux 2.6.12-1.1456_FC4 (localhost.localdomain)    10/27/05

Device:            tps   kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
hda               0.55         2.64         3.28    6631507    8231496
hdb               0.21         4.51         9.10   11342022   22855752

Device:            tps   kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
```

```
hda              0.00         0.00         0.00          0          0
hdb              0.00         0.00         0.00          0          0

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
hda              0.00         0.00         0.00          0          0
hdb              0.00         0.00         0.00          0          0
# 仔细看一下，如果是有侦测次数的情况，那么第二次以后所显示的数据，
# 则代表两次侦测之间的系统传输值！举例来说，上面的信息当中，
# 第二次显示的数据，则是两秒钟内(本案例)系统的总传输量与平均值。
```

透过 lspci 及 iostat 可以约略的了解到目前系统的状态啊！ 还有目前的主机硬件数据呢！知道这些信息后，我们就可以来玩一些比较不一样的东西啰！ ^_^

驱动 USB 装置：

在现在的计算机里面，你或许真的无法想象没有 USB 接口装置的主机～ 因为不论我们的键盘、鼠标、打印机、扫描仪、随身碟等等，几乎都是使用到 USB 来作为传输的接口的。 所谓这 USB (Universal Serial Bus) 最早是在 1994 年被发展出来， 到 1996 年前后发展出 version 1.0 ，当时的速度大约在 12Mbit/second， 到了 1999 年发展出 version 2.0 ，这一版的速度则提高到 480Mbit/second。

USB 有很多的优点啦，包括他是可以延伸的，每个 USB port 都可以最多接到 127 个装置！ 速度又快，又具有 Plug and Play (随插即用) 的优点，所以近期以来被用来作为携带式装置的主要数据传输接口呢！

- 关于 USB 的芯片版本：

目前 USB 的控制器主要有两种规格，分别是：

- OHCI (Open Host Controller Interface)：主要由 Compaq 所发展，包括 Compaq, SiS, ALi 等等厂商发展的芯片都是用这个模块；
- UHCI (Universal Host Controller Interface)：主要由 Intel 所发展， 包括 Intel, VIA 等等厂商发展的芯片都是使用这个模块。

但不论是哪一种 USB 的装置，这两种芯片都能够启动并且支持的。 不过，以使用上来说， UHCI 比较容易使用，但是他的驱动程序比较复杂一些， 可能也会消耗比较多的 CPU 资源就是了。

也就是说，基本上，如果你使用 lsmod 时，会发现到 (O|U)HCI 之类的模块时， 那就表示您的 Linux 主机已经有加载 USB 的驱动程序了啦！ 这也是目前 FC4 预设加载的模块之一啊！不过，如果您的 USB 装置比较多， 包括键盘、鼠标、游戏杆等等的 USB 装置时，那么可能还需要加载 USB Human Interface Device (HID) 模块才行呢！以鸟哥的 FC4 测试机来说，我的主机是 Intel 815 芯片的 (P-III 的 CPU)， 由于没有提供 USB 2.0 ，所以，鸟哥自己买了一张 USB 2.0 的扩充卡。然后，使用 lsmod 出现如下的画面啰！

```
[root@linux ~]# lsmod
Module                  Size  Used by
uhci_hcd               43345  0
ehci_hcd               48333  0
# 我仅列出这两个模块而已～上面的是 UHCI 模块，底下则是 USB 增强模块。
```

```
[root@linux ~]# modinfo ehci_hcd
....省略.....
description:    10 Dec 2004 USB 2.0 'Enhanced' Host Controller (EHCI) Driver
author:         David Brownell
license:        GPL
....省略.....
# 很有趣吧！说的是 Enhanced Host Controller 呢！
```

Tips:
事实上，更多的 USB 装置的信息都放置到 hotplug 这个套件里面呢～ 你可以到 /etc/hotplug.d/ 目录下去查阅一下相关的说明喔！



- 启动 USB 随身碟：

我们之前谈过 USB 的磁盘代号是： /dev/sd[a-p] 之类的，类似 SCSI 硬盘的代号， 这是因为 USB 的磁盘装置使用 SCSI 相关的装置代号，因此，如果您要使用 USB 随身碟的话，嘿嘿！那么您的 Linux 主机就得要支持 SCSI 装置才行～

此外，为了让 USB 磁盘装置顺利的被使用，因此，有时候还得要启动 usb-storage 模块才行～ 所以啰，光是有 USB 的 uhci 模块还不行，还得要配合 usb-storage 啦～ 而一般 USB 的装置都会被主动的侦测，核心也会主动的加载 USB 装置的驱动模块， 所以您应该不需要手动加载 usb-storage 才是。不过，如果老是无法驱动时， 那么不妨手动加载 usb-storage 试看看。

顺利加载各个需要的模块之后，直接下达 fdisk -l 应该就可以看到您的 USB 随身碟的装置代号才是！一般来说，如果是第一个 USB 磁盘装置的话， 应该可以看到一个名为 /dev/sda1 的装置，使用 mount 将他挂载起来即可啊！ 详细的挂载 (mount) 与挂载点还有档案系统格式 (filesystem) 鸟哥在这里就不谈了， 翻翻第二篇的内容去！ ^_^

- 启动 USB 打印机：

要驱动 USB 打印机也很简单啊！ 只要做好 USB 打印机的装置代号即可！反正我们的 usb 模块已经加载了嘛！ 要建立 USB 打印机的装置代号得使用 mknod 这个指令才行。 此外， USB 打印机装置的 major, minor 代号分别是 180 0， 所以，建立的方法为：

```
[root@linux ~]# mkdir -p /dev/usb
[root@linux ~]# mknod /dev/usb/lp0 c 180 0
[root@linux ~]# chown root:lp /dev/usb/lp0
[root@linux ~]# chmod 660 /dev/usb/lp0
[root@linux ~]# ls -l /dev/usb/lp0
crw-rw----  1 root lp 180, 0 Nov  7 16:03 /dev/usb/lp0
[root@linux ~]# echo "testing" > /dev/usb/lp0
```

鸟哥这里是以 FC4 为范例，要注意， /dev/usb/lp0 的权限必须要与 /dev/lp0 相同， 所以，我这里还得要使用 chown 与 chmod 来变换该装置档案的权限才行。 建立好之后，就可以使用 cups 之类的软件来

管理这一部打印机啰！^_^。事实上，除了比较早期的 Linux distributions 之外，较新的 distributions 已经帮我们建立好 /dev/usb/lp0 等档案装置了呢！真是方便！

在我们一般的生活当中，最常见的两种 USB 装置，就是随身碟与打印机了， 所以鸟哥在这里仅就这两种装置来介绍启动的方法，如果您还有其它的 USB 装置要驱动的话， 请参考底下这一篇的内容啊！

- http://www.linux-usb.org/USB-guide/book1.html

至于 USB 打印机的实际驱动，我们还得要继续 CUPS 章节的内容才行喔！

使用 lm_sensors 取得温度、电压等信息

玩计算机硬件的朋友们一定都听过所谓的『超频』这玩意儿，所谓的『超频』就是让系统原有的运作频率增加， 让 CPU/PCI/AGI 前端总线速度提升到非正规的频率，以取得较高的计算机效能。 这在早期对于单价还是很贵的计算机来说，可以让我们花比较少的钱去获得比较高效能的计算机哩！ 不过，超频要注意的地方可不少，包括电压不可高出 CPU 的负荷、CPU 风扇必须要强有力， 避免因为温度过高导致系统当机等等。

不过现今的计算机速度已经够快了，我们的 Linux 主机也实在不建议您超频， 因为整体效能可能增加不了多少，但是却会让您的主机寿命减少、系统不稳定呢！ 而由早期超频的『技术培养』过程当中，我们知道『CPU 的温度、系统的相关电压』 是影响主机是否稳定的一项重要指标喔！所以啰，如果能够随时掌握温度、电压， 其实对于系统还是有一定程度的监控啦。

其实各大主要主机板商与芯片组，都会有温度、压力的侦测器在主机内， 这个我们可以在主机板操作手册或者是在 BIOS 内的『 Monitor 』项目找到相关的温度、压力数据。 在 Windows 系统当中，厂商有推出相关的软件来侦测，那么在 Linux 当中呢？呵呵！也是有啊！那就是 lm_sensors 这套好用的东西了！

与之前版本不同的是，FC4 已经内建了这个 lm_sensors 套件了，所以我们不需要手动去安装他！ 真是好高兴啊～检查看看您的主机是否有这个玩意儿吧！

```
[root@linux ~]# rpm -qa | grep lm_sensors
lm_sensors-2.9.1-3.FC4.2
```

如果您的 Linux distributions 是比较早期的版本，那么就只好请您自行前往 http://www2.lm-sensors.nu/~lm78/ 官方网站直接下载 tarball 并且安装他啰～如果您使用的是 FC3 或 FC4 ，那么我们就直接来处理吧！

由于 lm_sensors 主要是依据『主机板芯片组的型号，带入相关的模块后，再侦测其温度、压力』的， 如果该主机板芯片组并不是 lm_sensors 所支持的模块，那自然就无法找出该芯片组的温压啰～ 所以啦，我们在使用 lm_sensors 之前，必须要确定主机板是有提供温度、电压的， 再来，必须要加载主机板的驱动模块，然后才有办法使用 lm_sensors 来进行侦测。

而，好消息是， lm_sensors 本来就提供我们一个不错的主机板芯片组侦测程序， 那就是 sensors-detect 这个指令。侦测到主机板芯片组后，将该信息写入设定文件当中， 就可以使用 sensors 指令直接读取目前的 CPU、机壳、电源、风扇等等的信息了！ 直接来作看看吧！

```
1. 先侦测主机板的芯片组啊!
[root@linux ~]# sensors-detect
# 开头会有一些简单的说明，看看就好!
It is generally safe and recommended to accept the default answers to all
questions, unless you know what you're doing.

 We can start with probing for (PCI) I2C or SMBus adapters.
 You do not need any special privileges for this.
 Do you want to probe now? (YES/no): y
Probing for PCI bus adapters...
Use driver `rivatv' for device 01:00.0: GeForce2 MX
Use driver `i2c-viapro' for device 00:07.4: VIA Technologies VT82C686 Apollo ACPI
Probe succesfully concluded.
# 接下来的行为当中，反正你就一直按 Enter  就可以了! 让他自动去侦测!

 I will now generate the commands needed to load the I2C modules.
 Sometimes, a chip is available both through the ISA bus and an I2C bus.
 ISA bus access is faster, but you need to load an additional driver module
 for it. If you have the choice, do you want to use the ISA bus or the
 I2C/SMBus (ISA/smbus)?

To make the sensors modules behave correctly, add these lines to
/etc/modules.conf:

#----cut here----
# I2C module options
alias char-major-89 i2c-dev
#----cut here----

To load everything that is needed, add this to some /etc/rc* file:

#----cut here----
# I2C adapter drivers
modprobe i2c-viapro
modprobe i2c-isa
# I2C chip drivers
modprobe eeprom
modprobe via686a
# sleep 2 # optional
/usr/bin/sensors -s # recommended
#----cut here----
# 上面的关键是重点! 告诉你要如何加载模块的一个简单的范例啊!
```

```
Do you want to generate /etc/sysconfig/lm_sensors? (YES/no):
Copy prog/init/lm_sensors.init to /etc/rc.d/init.d/lm_sensors
for initialization at boot time.

# 如果想要确认一下 sensors-detect 侦测到的结果是否正确，
# 可以使用 lspci 查阅一下喔！鸟哥的例子当中，是这样的：
[root@linux ~]# lspci
00:00.0 Host bridge: VIA Technologies, Inc. VT82C693A/694x [Apollo PRO133x] (rev c4)
00:01.0 PCI bridge: VIA Technologies, Inc. VT82C598/694x [Apollo MVP3/Pro133x AGP]
00:07.0 ISA bridge: VIA Technologies, Inc. VT82C686 [Apollo Super South] (rev 22)
# 确实找到的是 VIA 的芯片！没有问题！


2. 开始加载模块与修改一些设定数据文件啊！
[root@linux ~]# vi /etc/modprobe.conf
alias char-major-89 i2c-dev
# 将刚刚侦测到的模块给他写入到这个档案当中！这是 kernel 2.6 版的！
# 如果是早期的 kernel 2.4 核心，那就写到 /etc/modules.conf 当中！

[root@linux ~]# vi /etc/rc.d/rc.local
# Adding the sensor for VIA 686A Chip
/sbin/modprobe i2c-isa
/sbin/modprobe via686a
sleep 2
/usr/bin/sensors -s
# 虽然 sensors-detect 希望我可以加载四个模块，不过，鸟哥分析的结果，
# 发现我只要两个模块加载即可～所以，这里我有省去两个 modules 的载入喔！
# 当然啦！建议你还要看一看 /etc/sysconfig/lm_sensors 档案的内容，就比较更清楚了！

[root@linux ~]# sync; sync; reboot
# 虽然可以直接利用上述的指令在 bash 当中直接处理，不过我不喜欢这样，所以啰，
# 直接给他重新启动系统，就 OK 啦！另外，观察一下是否开机即启动 lm_sensors?
[root@linux ~]# chkconfig --list | grep lm_sensors
lm_sensors      0:off   1:off   2:on    3:on    4:on    5:on    6:off


3. 测试侦测主机的状况！
[root@linux ~]# sensors
via686a-isa-6000
Adapter: ISA adapter
CPU1 core: +1.73 V  (min =  +1.65 V, max =  +1.90 V)
CPU2 core: +1.73 V  (min =  +1.65 V, max =  +1.90 V)
I/O:       +3.42 V  (min =  +3.12 V, max =  +3.45 V)
```

```
+5V:       +5.18 V  (min =  +4.73 V, max =  +5.20 V)
+12V:     +12.24 V  (min = +11.35 V, max = +12.48 V)
CPU1 Fan: 4963 RPM  (min = 3000 RPM, div = 2)
CPU2 Fan: 5232 RPM  (min = 3000 RPM, div = 2)
CPU1 Temp: +39.7°C  (high =   +65°C, hyst =   +55°C)
CPU2 Temp: +40.4°C  (high =   +65°C, hyst =   +55°C)
SBr Temp:  +23.3°C  (high =   +65°C, hyst =   +60°C)
# 呵呵呵呵！不但可以看到温度、电压，还有风扇转速，实在太完美了！
# 咦！怎么会是双 CPU 啊！呵呵！因为鸟哥的主机板（这个测试的平台）是双 CPU 平台，
# 但是 lm_sensors 并没有帮我主动的分出双 CPU ，所以，鸟哥是自行进入设定档，
# 亦即是 /etc/sensors.conf 去进行修订的工作！
```

基本上，只要这样的步骤，您的主机就可以主动的侦测温度与电压，还有风扇转速等等信息。 不过，事实上，由于主机板设计的不同，所以侦测的结果很有可能是有误差的。以鸟哥的情况来说， VIA 694D 是适用双 CPU 的主机板，但是 lm_sensors 假定则是仅有单颗 CPU 的显示状况， 所以，显示的结果与实际的数据是有差异的～此时或许就需要进行调校了。 调校的步骤很简单，先确定使用 sensors 显示的结果每个项目代表的意义（可以参考 BIOS 硬件侦测结果的顺序来排列），然后进入 /etc/sensors.conf 进行修改即可。 鸟哥以自己的 via686a 这个芯片组来说明！

```
[root@linux ~]# vi /etc/sensors.conf
# 在这个档案当中，先找到你的主机板芯片组，参考其内容喔！
# 至于除了 chip 后面接的是芯片组外，其它的则以底下的格式来书写的：
# label  '实际侦测的输出或者是代号'  '使用 lm_sensors 输出的信息'
# 举例来说，第一颗 CPU 的侦测项目其实是 "2.0V" ，那个是实际存在
# 主机板当中的一个项目，至于后面的 CPU1 core 则是我们自己加上去的，
# 这样才可以在使用 sensors 时，看到输出的结果啊！因此，
# 那个 "2.0V" 不要动～动的是那个 "CPU1 core" 喔！

chip "via686a-*"
    label "2.0V" "CPU1 core"
    label "2.5V" "CPU2 core"
    label "3.3V" "I/O"
    label "5.0V" "+5V"
    label "12V" "+12V"
    label fan1  "CPU1 Fan"
    label fan2  "CPU2 Fan"
    label temp1 "CPU1 Temp"
    label temp2 "CPU2 Temp"
    label temp3 "SBr Temp"

# 至于底下的 set 则是在设定『最大、最小限制值』就是了～
# 其中，底下那个 in0 及 in1 分别代表第一、第二个 label 的意思，亦即是
# CPU1 core 及 CPU2 core 啦～再来，则是 temp1~3 ，
# 注意，如果项目不是 "2.0V" 这种以双引号取出的，则必须要使用原本的名称，
# 亦如 fan1, fan2 等等，这个务必了解才行！
```

```
set in0_min 1.65
set in0_max 1.90
set in1_min 1.65
set in1_max 1.90
set temp1_hyst 55
set temp1_over 65
set temp2_hyst 55
set temp2_over 65
set temp3_hyst 60
set temp3_over 65
set fan1_min 3000
set fan2_min 3000
```

修改完毕之后，就可以准备准备加入更新啰～那就是使用：

```
[root@linux ~]# sensors -s
```

确定更新了这个信息之后，再下达 sensors ，嘿嘿嘿嘿！就可以顺利的取得属于正确的信息啦！ 当然，如果想要以图表输出的话，那么不妨搭配 MRTG 来进行网页绘图～ 这部分网络上面文章就比较多一点，也可以先参考鸟哥的一篇旧文章：

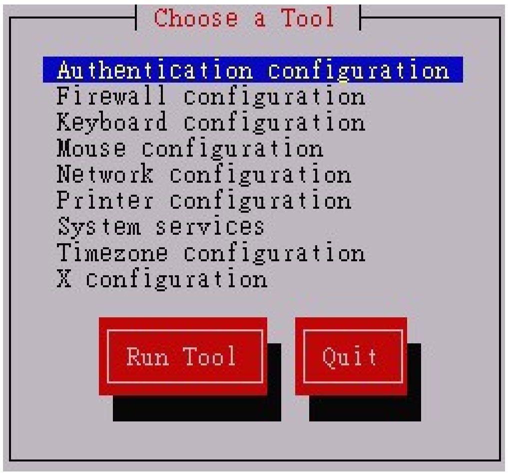
- http://linux.vbird.org/linux_security/old/04mrtg.php

加油啰！ ^_^

FC4 的系统操作： setup 功能

我们在基础篇谈到很多的概念问题，包括整个开机流程其实看的是 /etc/inittab 的规定项目， 每个服务启动的 scripts 是放在 /etc/init.d/scriptname 里面，但是启动设定档则可能会包含在 /etc/sysconfig 内。 举例来说，刚刚上面提到的 lm_sensors ，他的模块名称就放置在 /etc/sysconfig/lm_sensors 呢～ 当然，您也可以自行更改 /etc/init.d/lm_sensors 的内容，让他不要去读取 /etc/sysconfig/lm_sensors， 但毕竟是系统预设的状态比较好控制啊～

在我们了解到这些基本的设定信息后，呵呵～终于可以用一下系统提供给我们的简单的操作接口啦～ 在 Fedora 里面，有个 Red Hat 系统总是有提供的工具，套件名称为 setuptool ， 整个执行的过程则直接下达 setup 就能够处理哩～真是好方便呐！ 有点类似窗口就是了。不过，没有很建议您使用，因为其实直接使用指令或 vi 也可以达成 setup 所提供的功能啦！ ^_^。当你以 root 的身份下达『 setup 』后，会出现这样的咚咚：(当然啦！您必须要安装 setuptool 套件才行！ 而且，我是使用 LANG=en_US 来进行画面撷取的，如果您是在可以显示中文的环境下， 可以使用 zh_TW.big5)



图、Red Hat 系统的 setup 指令

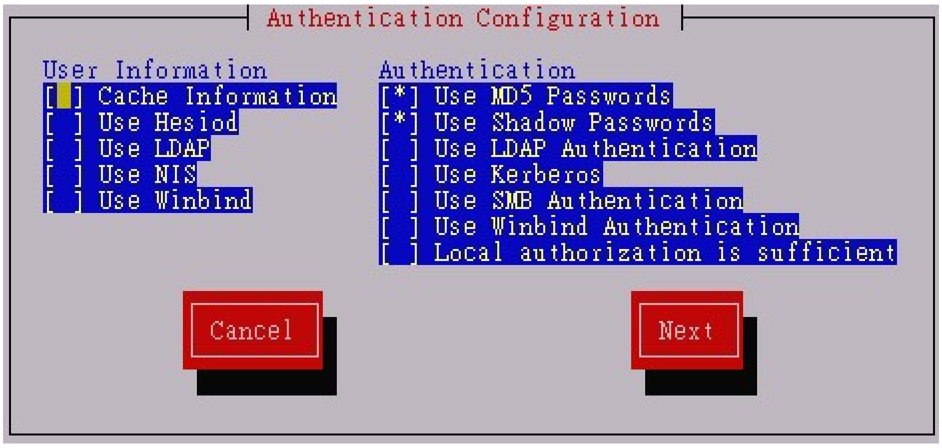
主要的项目其实就是底下这些数据啦：

- Authentication configuration：
  这是与使用者认证机制较有相关的设定数据， 包括认证来源的主机确认与本机数据的决定等等；
- Firewall configuration ：
  这个是关于系统的防火墙设定。一般来说，手动设定比较好，用这个东西设定， 常常会搞不懂他到底在哪些地方作了哪些事情～不容易搞定啦！
- Keyboard configuration ：
  包括键盘的形式与最重要的键盘的按键对应表。注意， 这个设定仅与 tty 接口有关，至于 X Window 则不是以这个为设定值。
- Mouse configuration ：
  设定鼠标的型态的地方啦；
- Network configuration ：
  设定网络参数的地方，包括 IP, network, netmask, dns 等等，不过， 还需要看完服务器篇关于网络基础的介绍后，才能够比较了解设定值的意义啦！
- System services ：
  其实就是使用 ntsysv 的内容喔～亦即设定一些系统服务的地方；
- Printer configuration ：
  设定打印机啰；
- Sound card configuration ：
  若您的主机上面有声卡，这个项目才会出现！就是设定声卡啦！
- Timezone configuration ：
  设定时区；
- X configuration ：
  设定 X-Window 相关的硬件设定！亦即是设定 X server 啦！ 也可以手动修改 /etc/X11/xorg.conf 或 XF86Config 啊！

底下我们就一个一个来介绍这玩意儿吧！并且请注意，你可以利用 <tab> 按键移动， 也可以利用上下按键移动，利用空格键或者是 [enter] 按键来决定是否需要！

• Authentication configuration

在按下了 Authentication configuration 后，会出现如下画面：

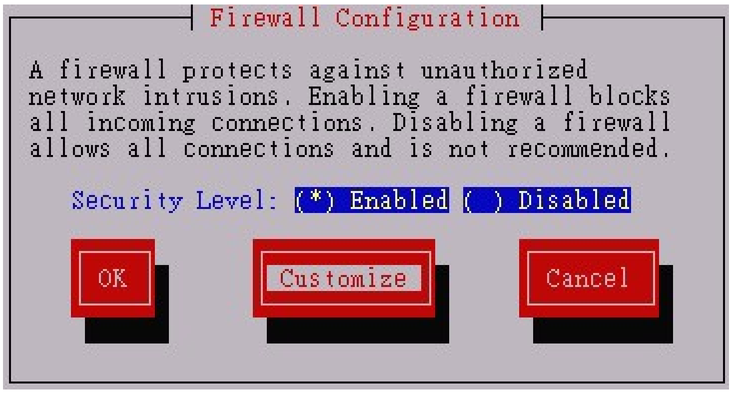


图、Setup 的认证机制部分

关于使用者身份认证上面，Linux 提供了相当多的机制喔！包括 Linux 本机系统上面的 MD5 密码编码，以及利用 shadow 将密码移动到 /etc/shadow 档案当中等等。当然啦， 我们也可以利用网络上面的身份认证主机，包括 NIS, LDAP 以及 Windows 系统提供的认证主机等等， 只不过这些网络主机提供的服务呢，我们都得要进行客户端联机的设定，并不是启动就会生效的！ 所以啦，我们仅有本机密码档案而已，当然只要选择这两个数据就够了。另外， 这个设定数据主要是修改 /etc/sysconfig/authconfig 档案哩！ 你也可以自行手动修改该档案即可。

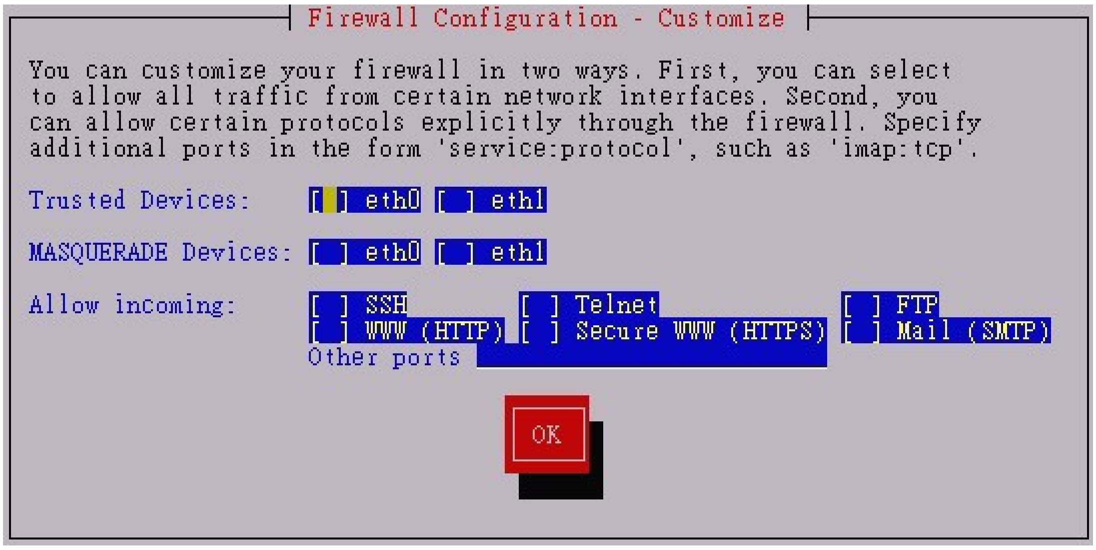
• Firewall configuration

按下 Firewall configuration 后，会出现如下画面：



图、Setup 的防火墙部分

注意一下，由于我们在安装的时候，并没有选择防火墙，因此，这里预设会是 (Disabled) 那一项， 如果你想要启动防火墙，那么就得要先选择 Enabled 后，将光标移动到 Customize 后， 才能够决定你想要服务项目。按下 Customize 后，会出现如下画面：



图、Setup 的防火墙部分

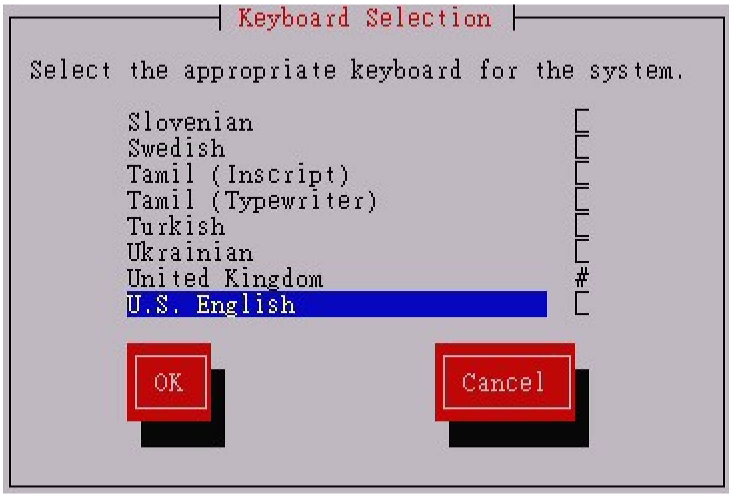
这个地方不是三言两语讲的完的！包括信任网域，以及允许进入的服务器封包～ 很是麻烦。基本上，你只要这样想就好了：

- Trusted Devices：这是信任网域， 如果你有两张网络卡，一张是 eth0 对内，一张是对外，假设是 eth1 ，那么如果你想要让 eth0 的进出封包都是为信任，那么这里就可以将 eth0 勾选。不过，要非常非常注意， 接到外部网域 (Internet) 的那张网络卡，千万不能勾选，否则就挂了！
- MASQUERADE Devices：这个是『封包伪装』的功能， 亦即是进行 IP 分享器的功能啦！如果你的 Linux 主机是作为类似 IP 分享器的功能， 那么对外那张网络卡就得要启动 MASQUERADE 才行！以上面的例子来说，就是勾选 eth1 啦！
- Allow incoming：这里提到的就是各个服务的内部项目， 举例来说，你的 Linux 有提供 WWW 服务，又希望大家都能够来查阅， 那么这个时候就可以在 WWW 那个项目前面勾选啦！

基本上，这个动作仅是在建立 /etc/sysconfig/iptables 这个档案而已。而这个档案预设是不存在的 (因为我们没有启动防火墙啊！) 。 这里你先有个概念即可，因为，我们未来会介绍以 shell script 的方式建立属于您自己的防火墙系统， 细节我们会在服务器篇慢慢作介绍的啊！

- Keyboard configuration

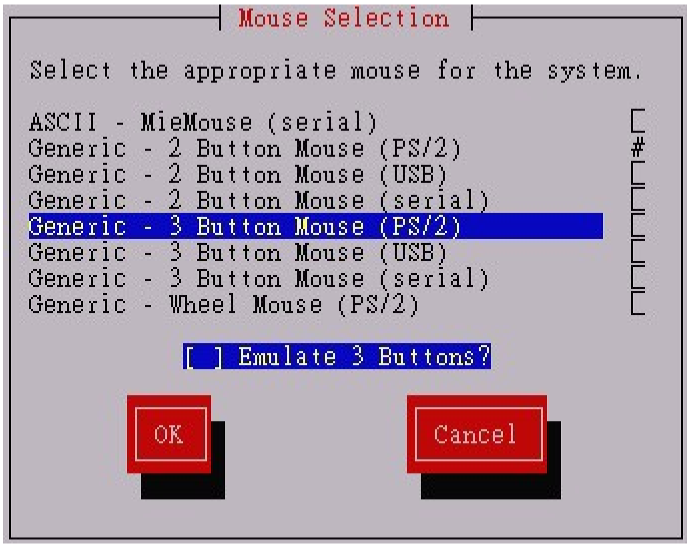
这个画面如下：基本上，其实就是选择键盘按钮的对应表而已啦！ 这个设定会修改 /etc/sysconfig/keyboard 就是了。



图、Setup 的键盘选择部分

- Mouse configuration

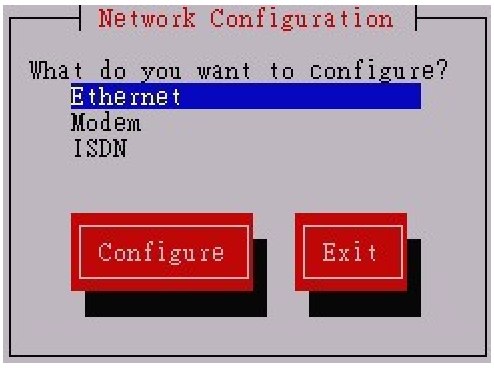
请选择您的鼠标类型啊！其实直接修改 /etc/sysconfig/mouse 也是可以啦！



图、Setup 的鼠标选择部分

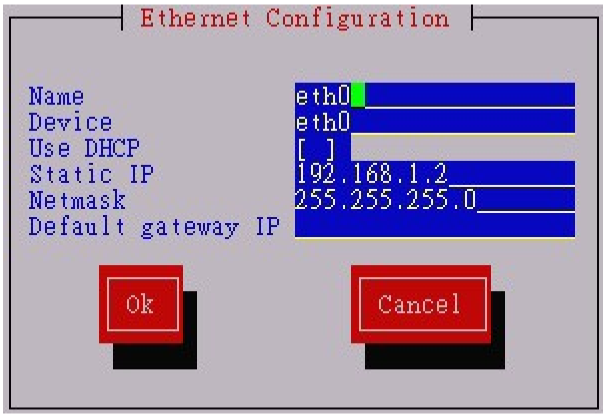
- Network configuration

在网络的部分，由于我们的 Linux 支持的网络联机方式众多，而我们台湾地区比较常使用以太网络 (Ethernet) 的架构，如果你有网络卡，那么别怀疑，当然是选择 Ethernet 啰～ 如下图所示啦！



图、Setup 的网络选择部分

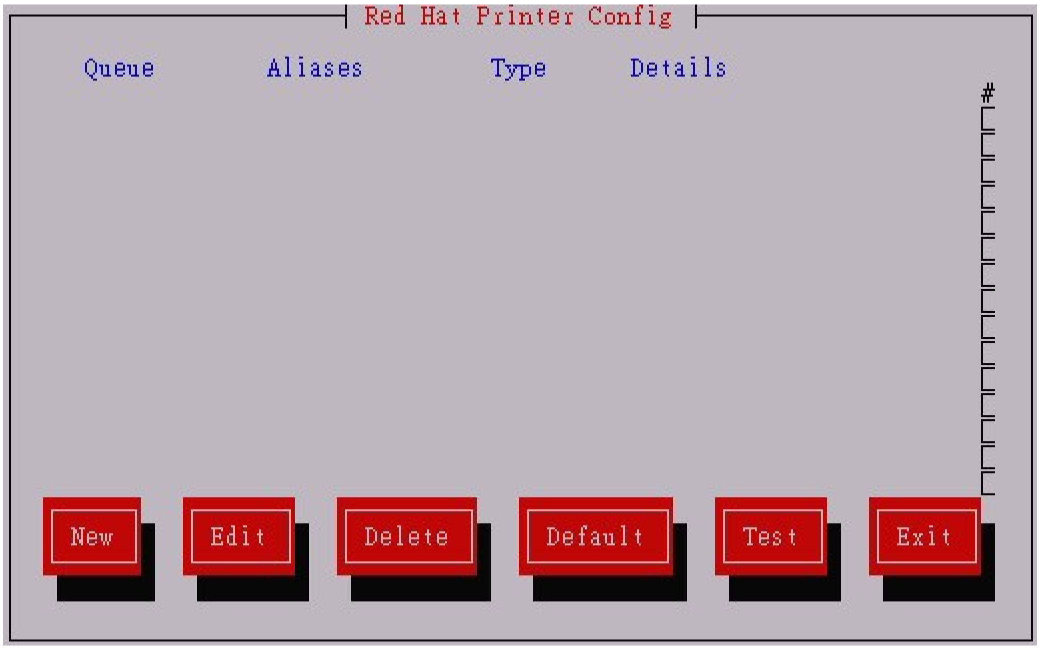
选择完毕之后，会出现如下窗口，这个就比较麻烦了～基本上，仅有两种格式。 第一种，你可以直接勾选『Use DHCP』那个项目，让类似 IP 分享器自动的帮你设定好 IP； 另一种，就如同下列图样，不要勾选 DHCP，直接给予一个 IP 以及相关网络参数即可。 这些网络参数的设定我们会在服务器篇再介绍的啦！



图、Setup 的网络选择部分

- Printer configuration

这个项目在决定你的打印机类型啦！基本上，他更动的就是 FC4 提供的 CUPS 打印机队列相关信息～ 而且我们等一下整个设定的项目，其实都会写入 /etc/cups/ 目录下的许多档案当中哩～ 这个功能对于建立打印机的动作来说，实在是相当的不错喔！ 整个动作很繁琐，我们一个一个来说明吧！按下 printer configuration 后，出现这样：



图、Setup 的 Printer 选择部分

由于我们需要新增打印机，因此，当然选择那个 new 啊～这里请注意一下， 我们仅是作个测试而已，所以设定错误也没关系的啦～我这里假设我有一部 HP 2300 Laserjet 的打印机， 而且是接在本机上面的 USB port ，所以就进行这个动作：



图、Setup 的 Printer 选择部分

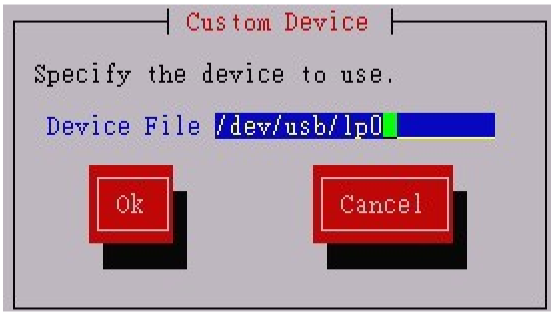
上面的动作仅是在设定一个打印机的命名而已，那个 hp2300 是随便自己命名的， 无所谓，但是下方那五个选项就重要了！由于我们使用的是本机装置，因此，就得选 LOCAL 那一项才行。 如果你有特殊需求，那么才自行设定吧！万一是网络打印机呢？有 IP 的打印机， 那么这里依旧选择 LOCAL 喔！不要搞错了～

接下来，按下 Next 之后，出现底下的画面，事实上，如果 Linux 主机核心有侦测到打印机， 那么底下画面的中间部分就会显示出该打印机的相关装置项目。不过，我们是测试嘛！ 当然也会侦测不到啊！因此，这里我们要选择 Custom 那个项目(自订的意思)。



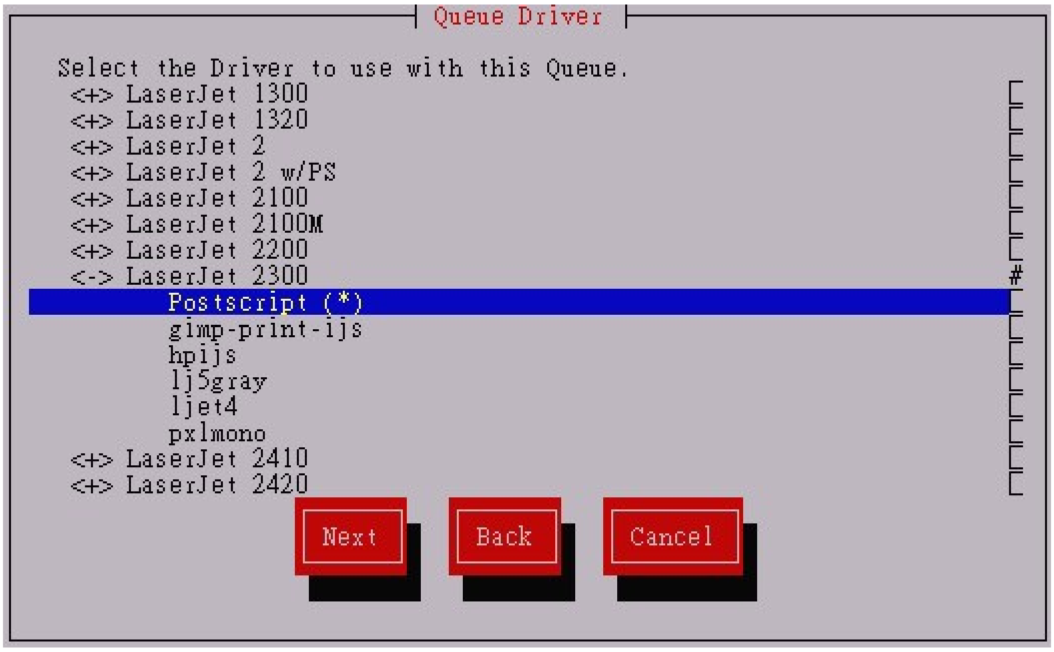
图、Setup 的 Printer 选择部分

然后就是填写打印机接在主机的那个连接端口上头啦！我这里假设是 USB 打印机，所以自然就是 /dev/usb/lp0 ，如果是 25 针串行端口的打印机，就填 /dev/lp0 。如果你没有 /dev/usb/lp0 这个装置档案， 参考上面我们提到的 USB 的装置建立方法吧！



图、Setup 的 Printer 选择部分

接下来才是选择打印机的型号啊！你可以按上下键，配合 enter 按键来选择到你的打印机。 我们这里使用 Postscript 打印格式来做为打印机的主要打印模式啊！

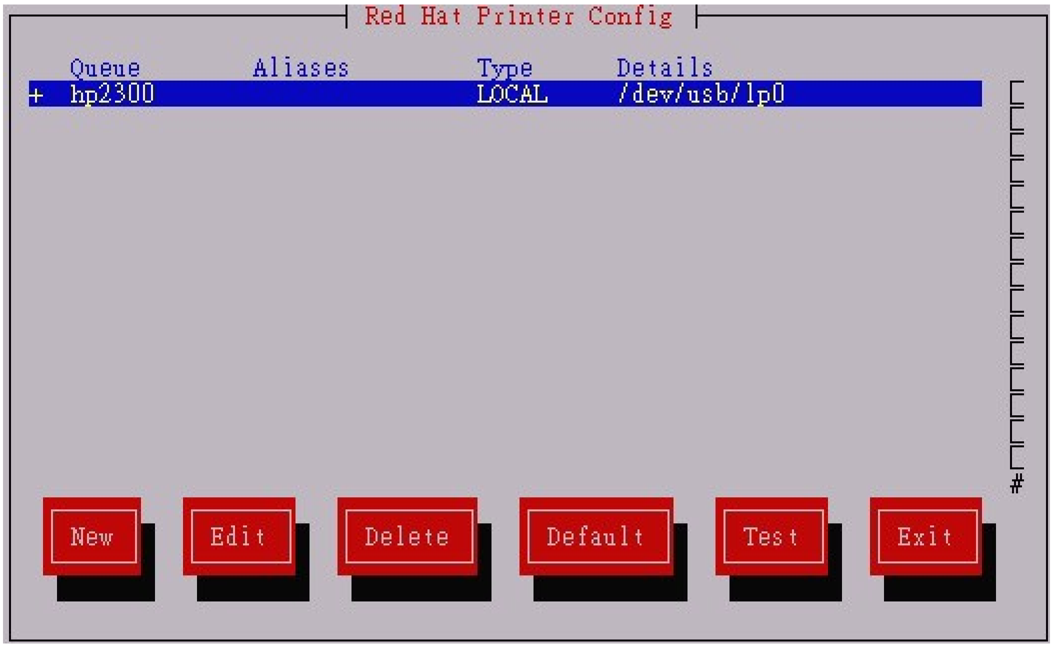


图、Setup 的 Printer 选择部分

最后有个总结的部分，看看就好～最后就会得到一个名为 hp2300 的打印机了！

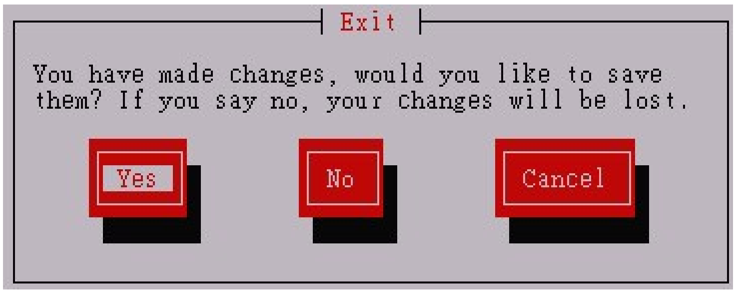


图、Setup 的 Printer 选择部分



图、Setup 的 Printer 选择部分

最后，当你要离开打印机的设定时，他会显示是否要储存的动作，当然选择『Yes』啊！ 不然设定可不会被记录下来的喔！



图、Setup 的 Printer 选择部分

其实打印机在 Linux 上面设定是挺麻烦的，尤其是您必须要清楚的了解到打印机使用的模块为何？ 所以，基本上，使用这个项目来设定好你的打印机，实在是一个不错的方法。 而刚刚的动作做完后，其实会有几个数据被更动，分别是：

- /etc/cups/printers.conf：主要是打印机的相关设定项目；
- /etc/cups/cupsd.conf：使用 printer 权限的设定项目；
- /etc/cups/ppd/hp2300.ppd：就是刚刚我们选择的打印机型号的模块。

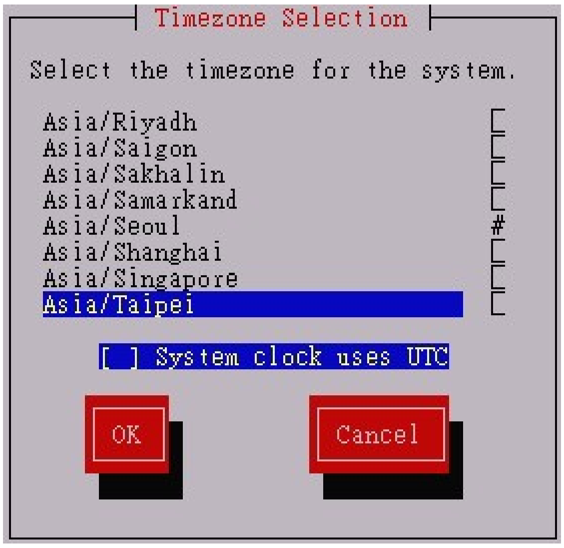
更多与打印机有相关的数据，我们在底下的 CUPS 章节再来讨论啰～

- System services

基本上，这个就是 ntsysv 啦～这个我们已经在 认识系统服务 当中提过了，所以就不再浪费章节啦！

- Timezone configuration

时区的设定，其实就是找出与 /etc/sysconfig/clock 有关的设定项目而已。 实际上，下面图示出现的咚咚，就是在 /usr/share/zoneinfo/ 有关而已。



图、Setup 的时区选择部分

- X configuration

说穿了，其实就是直接设定 X Server 的设定档，在 Xorg 就是 /etc/X11/xorg.conf 啦！ 至于整个结果有没有成功，还得要观察 /var/log/Xorg.setup.log 这个档案的输出结果才能了解喔！

鸟哥个人认为，这个 setup 的工具是很好用的～只是，如果能够完全清楚整个系统架构的话， 再来玩这个小程序会比较好啦！ ^_^。当然啰，打印机的工具实在很不错～ 可以提供给您参考参考！

利用 CUPS 架构您的打印机：

打印机对于日常生活来说，很重要吧！呵呵～没错啊！尤其我们的 Linux 主机如果未来还要作为 Printer server 的话，那么自然就得要先建立好打印机的联机啦！刚刚前一个小节我们仅是简单的利用 setup 来建立一部不存在的打印机而已～基本上，你必须要了解到整个 Linux 针对打印的动作是如何， 才能够清楚的了解到我们要如何管理打印机啊！ ^_^

- BIOS 当中，针对打印机的设定项目：

在打印机的设定当中，鸟哥曾经发生过一件相当糗的事情，我拼了老命的设定 LPRng 以及 CUPS 就是无法设定好 Printer ，虽然已经连上打印机了，但是就是印出来都是乱码， 很伤脑筋～等到我花了一整天去恶搞之后，最后竟然发现，错误的地方在于『BIOS 的并行端口设定』唉！ 伤脑筋的很～还记得每次开机的时候系统都会去读取 BIOS 的设定吗？一般来说，按下 DEL 按键后，会进入 BIOS 的设定画面，在该画面当中，选择相关的设定参数，与 Parallel 有关的项目，将『模块』改成 EPP/SPP 吧！这样就可以支持您的打印机了！天呐！ 就因为这个设定值，让鸟哥花了一两天的时间，还差一点将主机砸掉....

- Linux printer 的打印工作之进行：

在 Linux 环境下，如何开始一个打印工作呢？基本上，打印工作当然是由程序所产生的啦～ 不论是 Open Office 之类的办公软件，或者是直接以 lpr 来进行打印的工作，反正，总是需要有打印的指令后， 产生一些信息，交给打印机来处理就是了。在进入打印机的实际打印之前， Linux 会先将该项打印工作放置到队列 (queue) 当中，而每一项工作都会被分为两个档案， 分别是实际要打印的数据，以及该打印工作的权限啊！。最后由 Linux 所支持的打印模式 (LPRng 或 CUPS) 来将该数据转成打印机认识的格式后， 就可以由打印机输出了。等到打印完毕，该工作就会被 queue 所移除。

就是因为如此，因此，我们虽然建立了一个不存在，或者是打印机暂时无法使用的情况下， 依旧是可以使用打印软件来打印数据的，只是该打印工作会被暂时存放在 queue 当中就是了。 另外，那个打印模式是很重要的一个概念喔！ 打印机通常仅认识自己的数据格式，所以，我们必须要让 Linux 将数据处理成打印机能够读取的数据格式啊！ 一般来说，Linux 里面有个很通用的打印信息，亦即是 Postscript 打印格式， 但是您的打印机可不见得会支持这种格式。果真如此时，就得要使用所谓的滤镜 (filter) 来处理数据成为打印机认识的格式了。

由于 postscript 的打印比较好，因此，实在很建议您， 直接购买支持 postscript 打印格式的打印机就好了。那么如何得知您的打印机是否支持该格式？ 建议您可以前往这里查阅一下啰：

- http://www.linuxprinting.org/

举例来说，我们研究室使用的是 HP Laserjet 2300 的机型，所以鸟哥选择了上面网页当中的 Printer list，亦即是：http://www.linuxprinting.org/printer_list.cgi ，然后选择 『HP』与『LaserJet 2300』后，出现如下网页的连结：
http://www.linuxprinting.org/show_printer.cgi?recnum=HP-LaserJet_2300 ， 在该网页当中，他有提到 Postscript 的格式在这部打印机上面可是工作的很完美喔！ 呵呵！这样就对啦！ ^_^

- Linux Printer 的 daemon

Linux 管理打印机的 daemon 主要分为两种，一种是 LPRng ，一种则是较新的 CUPS (Common Unix Printing System)。我们这里仅就 CUPS 来进行说明啰。

刚刚提到，我们的打印工作是由程序达成的，例如 lpr 等指令。 而打印工作被建立后，则被放置到队列当中等待 Linux print daemon 的分析与转换。这个 CUPS 的 daemon 就是在进行这个数据转换的工作啦！

CUPS 除了可以利用 /etc/cups/mime.types 辨识待转换打印数据的格式之外， 他还可以使用不同的滤镜 (filter) 来转换格式，相关的滤镜都放置到 /usr/lib/cups/filter/ 目录下。 当然啦，我们提到有个 postscript 打印格式吧，是否为 postscript 格式有不同的转换方法喔：

- Postscript 格式
  由于 Linux 对于打印机格式最熟悉的就是 postscript 了，因此，如果是 postscript 的打印机，那么打印数据将会被 /etc/cups/mime.conves 及 /usr/lib/cups/filter/pstops 转换与分析页数，然后直接交给打印机将数据输出！
- 非 postscript 格式
  当您的打印机对 postscript 格式并不支持时，那么该数据便会以其它的滤镜进行数据格式的转换。 转换的数据可能是 Ghostscript 格式，或者是直接以打印机的格式来直接打印。 无论如何，您还是得要了解一下打印机的打印格式才行啊！

为了要让数据变成 postscript ，好让打印机能够顺利的打印，因此， 我们的 cups 会去参照所谓的 Postscript Printer Discription (PPD)， 亦即是 postscript 的打印机描述定义数据，并且依据该定义来将数据转换成为 postscript 的格式啊！

事实上， CUPS 已经帮我们建立了很多 PPD 定义档了，透过这些 PPD 档案的描述， 我们的 CUPS 可以自行取得适用的滤镜，不再需要像以前的 LPRng ，还得要自行测试滤镜是否工作成功说！这些适用的 PPD 都被放置到底下的目录当中了：

- /usr/share/cups/model

但是，由于这些定义数据都可能一直在更新，因此，我们的 FC4 使用的是利用 foomatic 这个套件来辅助进行在线驱动程序的下载呢！所以啊，如果您的 Linux 没有连接上 Internet 的话，那么可能就无法取得最正确的打印机描述定义文件 (PPD) 了。 另外，在鸟哥看过的 Distributions 当中， SuSE server 9 就直接将整个打印机的 PPD 由 http://www.linuxprinting.org/download/PPD/ 捉下来， 呵呵！也是可以啦!所以说,如果您想要自行下载最新的打印格式定义档案，那就自行到 http://www.linuxprinting.org/ 去搜寻属于您的打印机 ppd 档案， 然后将他放置到 /usr/share/cups/model/ 当中即可！

- CUPS 支持的联机模式

在一般内部局域网络当中，较常见的打印机连结方式与分享方式有底下这几种：

- socket
  数据透过 internet socket(端口口)来传送，一般为 port 9100 或 35。 如果想要进行数据的传输与打印，可以透过在浏览器上面输入： socket://host-printer:9100/ 来进行。不过，这种模式不常用就是了。
- LPD (Line Pritner Daemon)
  就是我们之前提到的 LPRng 所支持的主要 daemon 啦～他是较早的打印协议啊， 主要是利用串行端口来达成打印的需求，打印机名称就是 LPT1/LPT2... 等等。 目前还是可以在比较早期的 Linux distributions 看到这种打印方式。
- IPP (Internet Printing Protocol)
  这是目前比较流行的打印机打印协议，我们的 CUPS 预设也是支持这种协议啊！ 当启动 IPP 时，打印机会启动 port 631 ，打印的数据就是透过这个 port 来进行传送的。另外，如果您的打印机或者 Linux 主机启动了 ipp 之后， 嘿嘿！你可以直接使用浏览器，输入：
  ipp://printer_IP/printername 就能够直接在线处理打印机的设定了！方便的很啊！
- SMB (Standard Message Block)
  也有称为 Server Message Block 的，那是什么啊？说穿了，不就是网络上的芳邻吗？ 没错啦！就是利用网芳提供的打印机来进行打印的意思！ 协议使用的是：
  smb://user:password@host/printer 。

我们 FC4 预设的 CUPS 使用的就是 IPP 这个协议说～也就是说，我们的 Linux 如果能够顺利的接上 printer 的话， 那么他就是一部网络打印机了啦！就这么简单啊～那么我们的 CUPS 预设可以支持哪些连接接口呢？ 有底下这些啦：

- parallel ：平行串行端口啊，就是 25 针那种玩意儿！他是连接到 /dev/lp[0-2] 等装置。 在 CUPS 里面的装置使用格式为： parallel:/dev/lp0;

- Network Printer ：网络打印机，例如 HP LaserJet 2300 就内建有网络卡， 也就是说，这个打印机是具有 IP 的，那么在 CUPS 内的使用格式为： ipp://hostname_or_ip/printername；
- USB ：一般越来越常见的 USB 打印机啊！ CUPS 使用的格式为： usb:/dev/usb/lp0 。

- 在 FC4 下启动 Printer

在 FC4 底下要启动打印机的话，相当的简单啊！其实只要照着刚刚我们上面提到的 setup 指令内的 printer configuration 的步骤，一步一步的给他设定下去，就 OK 了啦～但如果你的系统并不是 FC4 的话， 怎么办啊？没关系～我们可以手动来建立 CUPS 所需要的所有资料啊～ 更多的详细信息可以参考 cups 套件提供的 documents 或者是 man page 说～ (rpm -ql cups) 。

在底下，鸟哥以我们研究室的 HP LaserJet 2300 这一部打印机作为介绍， 这部打印机本身具有网络卡，他的 IP 假定为 192.168.10.119 ， 接下来，就开始来设定吧！

---

1. 下载适合的 ppd 定义档：

   前往 http://www.linuxprinting.org/printer_list.cgi 输入打印机的型号， 然后在出现的画面当中，选择『Recommended driver: Postscript (Home page, custom PPD)』 里面的那个『custom PPD』，给他点下去，将那个档案捉到 /usr/share/cups/model 里面去就对了！ 以鸟哥的情况来看，刚刚利用我查到的网页连结数据，可以这样做：

```
[root@linux ~]# cd /usr/share/cups/model
[root@linux model]# wget http://www.linuxprinting.org/foomatic-db/db/
source/PPD/HP/mono_laser/hp_LaserJet_2300.ppd
[root@linux model]# ls -l
-rw-r--r--  1 root root 55288 Oct 24 00:52 hp_LaserJet_2300.ppd
```

   这就是等一下我们要使用的模块。请注意，在 cups 使用的模块上面， 都是被放置到 /usr/share/cups/model 内！不要随便摆放～因为我们后面要使用的指令， 会主动读取这个目录内的主要 PPD 定义档啊！

---

2. 确定 CUPS 及打印机已启动：

   这个设定很简单啊！这样做就对了：

```
[root@linux ~]# /etc/init.d/cups start
[root@linux ~]# netstat -tlunp | grep 631
tcp  0  0 127.0.0.1:631     0.0.0.0:*    LISTEN    7228/cupsd
udp  0  0 0.0.0.0:631      0.0.0.0:*              7228/cupsd
[root@linux ~]# nmap 192.168.10.119
```

```
Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2005-11-11 14:15 CST
Interesting ports on hp2300 (192.168.10.119):
(The 1652 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
21/tcp   open  ftp
23/tcp   open  telnet
80/tcp   open  http
280/tcp  open  http-mgmt
443/tcp  open  https
515/tcp  open  printer
631/tcp  open  ipp
9100/tcp open  jetdirect
# 嘿嘿！远程的 Printer 确实是有提供 631 这个 port 的！
# 假如您具有 USB 打印机的话，那么应该就要变成这样：

[root@linux ~]# echo "Testing printer" > /dev/usb/lp0
# 如果打印机有数据输出，那么就表示联机没有问题啊！
```

由于不同的打印机联机模式会产生不一样的测试方法， 所以啰，如果是接在本机上面的，直接使用最简单的数据流重导向就可以测试了！ 如果是网络打印机，那就直接测试网络联机与使用 nmap 来进行测试啦～ 那个 nmap 我们会在服务器篇在好好的谈～现在只要知道他可以用来侦测某部主机的开启的 port， 算是一个简单的黑客软件～所以，不要拿 nmap 来侦测别人的主机！会有司法问题喔！

---

3. 开始加入打印机到 CUPS 当中：

要加入打印机让 CUPS 来管理，可以使用刚刚提到的 setup ，不过，这个工具似乎无法加入网络打印机的样子！ 所以，如果像鸟哥这种网络打印机的话，可能就不可以使用 setup 啦～这个时候，就使用那个好用的 lpadmin 来进行管理啰！

```
1. 加入打印机到 CUPS 当中：
[root@linux ~]# lpadmin -p [pritner 队列] -v [装置代号] -m [model] -E
参数：
-p  ：后面接打印机的名称，注意，这个名称是你自订的，可以随便取一个你记得住的名字
-v  ：后面接的是装置代号，可以是本机的串行端口或者是 USB ，当然也可以是网络打印机
      串行端口    ：   parallel:/dev/lp0
      USB        ：   usb:/dev/usb/lp0
      网络打印机：    ipp://192.168.10.119/
-m  ：就是刚刚提到的那个 ppd 定义档。这个档案必须要被放置到
      /usr/share/cups/model 当中，然后以该目录作为相对路径来书写文件名喔！
-E  ：作为接受 (enable) 打印工作的意思
```

```
2. 设定预设或删除打印机
[root@linux ~]# lpadmin [-xd] [printer 队列]
参数：
注意， -x 与 -d 不能同时使用，另外， printer 队列 必须是已存在的打印机名称
-x  ：删除一个在 CUPS 管理的打印机
-d  ：若有多部打印机存在 CUPS 当中，使用 -d 可以指定一部预设的打印机。
      则当我们在打印时，忘记选择打印机型式，则以此部预设打印机来打印。

范例一：在本例中，加入一部网络打印机，我的打印机队列填入 laserjet2300
[root@linux ~]# lpadmin -p laserjet2300 -v \
> ipp://192.168.10.119/ -m hp_LaserJet_2300.ppd -E
# 那个 laserjet2300 是自己设定的，怎么设定都没关系啦！
# 至于 hp_LaserJet_2300.ppd 则是我们刚刚由网站下载的档案，
# 注意，一定要放在 /usr/share/cups/model/ 目录当中喔！注意注意！

范例二：加入一个 usb 的打印机喔！
[root@linux ~]# lpadmin -p laserjet2300 -v \
> /dev/usb/lp0 -m hp_LaserJet_2300.ppd -E

范例三：让 laserjet2300 这部打印机成为预设打印机！
[root@linux ~]# lpadmin -d laserjet2300

范例四：删除 laserjet2300 这部打印机：
[root@linux ~]# lpadmin -x laserjet2300
```

其实刚刚这个 lpadmin 的动作，只是在更新 /etc/cups/ 目录里面的两个数据而已， 一个是 /etc/cups/printers.conf ，这个档案主要是规范了打印机的相关装置、是否接受打印工作、 打印机的队列名称、页面的限制等等，反正就是整个打印机的规范就是了。 至于这个打印机相关的 PPD 档案则是以打印机的队列名称连结到 /etc/cups/ppd/ 目录下， 以上表的范例一中，我们定义出 laserjet2300 这个队列，使用的是 hp_LaserJet_2300.ppd 这个定义档，则你会发现， /etc/cups/ppd/laserjet2300 与 /usr/share/cups/model/hp_LaserJet_2300.ppd 是相同的喔！因为只是要作为一个队列打印机的对应之用嘛！

```
[root@linux ~]# cat /etc/cups/printers.conf
# Printer configuration file for CUPS v1.1.22rc1
# Written by cupsd on Fri 11 Nov 2005 02:40:01 PM CST
<DefaultPrinter laserjet2300>
Info laserjet2300
DeviceURI ipp://192.168.10.119/
State Idle
Accepting Yes
JobSheets none none
```

```
QuotaPeriod 0
PageLimit 0
KLimit 0
</Printer>
# 看到否？刚刚我们使用 lpadmin ，增加的信息都在这里啊！
# 重点其实是那个 DeviceURI 及 Info 这两个玩意儿啦！重要的很！
```

---

4. 开始打印测试

我们可以使用 lp 或者是 lpr 来进行打印的工作～试看看先！

```
[root@linux ~]# lpr [-E] -P [printer 队列] -# [打印份数] -U [username] file
参数：
-E  ：是否加密的意思～一般不需要这个参数；
-P  ：如果你有两部以上的打印机，想要在不同的打印机上面打印，
      就需要使用 -P 来选择啊！
-#  ：如果你要一次打印多份文件，用这个 -# 加上份数就对了！
-U  ：有些打印机有限至使用者，此时就得要使用这个参数；
范例：

范例一：
[root@linux ~]# lpr -P laserjet2300 /etc/passwd
```

能不能打印除了 cups 的设定之外，打印机是否有设定防火墙也是有关系的喔！ 鸟哥最近一次在进行测试时，不知道为了什么，打印机突然无法接受我 Linux 端的封包要求， 鸟哥就给 Printer 整个重新开机，咦！突然又好了～ 可能是....打印机也需要休息吧？？ @_@

```
[root@linux ~]# lp -d [printer 队列] -n [打印份数] file
参数：
-d  ：后面接的是打印机的队列名称。如果有多部打印机才需要指定；
-n  ：就是打印的份数啊！
范例：

范例一：打印出 2 份 /etc/issue 数据
[root@linux ~]# lp -d laserjet2300 -n 2 /etc/issue
```

如果想要了解整个打印的信息，与整个打印机的状态，可以使用底下的指令啊～

```
1. 了解打印机的目前状态
[root@linux ~]# lpstat [-adprt]
参数：
-a  ：列出目前可以接受打印工作的打印机队列名称；
-d  ：列出目前系统的预设打印机；
-p  ：列出每部打印机的接受工作的状态，包含工作的 ID；
-r  ：列出目前 CUPS 是否有在运作？
-t  ：列出较为详细的打印机信息啊！
范例：

范例一：目前的预设打印机，与系统上面可以接受打印的打印机为：
[root@linux ~]# lpstat -a
laserjet2300 accepting requests since Jan 01 00:00
[root@linux ~]# lpstat -d
system default destination: laserjet2300

范例二：列出目前系统上面所有与打印机有关的信息？
[root@linux ~]# lpstat -t
scheduler is running
system default destination: laserjet2300
device for laserjet2300: ipp://192.168.10.119/
laserjet2300 accepting requests since Jan 01 00:00
printer laserjet2300 is idle.  enabled since Jan 01 00:00
        Print file accepted - job ID 3.



2. 打印工作的观察
[root@linux ~]# lpq [-al]
参数：
-a  ：列出所有打印机上面在队列当中的工作情况；
-l  ：用其它较长格式来输出打印的相关信息 (拥有者与档案大小等等)
范例：

范例三：显示出目前所有打印机的工作队列状况
[root@linux ~]# lpq -a
Rank    Owner   Job     File(s)                         Total Size
1st     root    3       passwd                          4096 bytes
# 上面的意思是，有一份工作，该工作是打印出 passwd 那个档案，
# 这个工作的号码是 3 号 (Job)，该工作的建立者为 root 。
# 这是个很重要的地方，因为该项目仅有 root 可以控制～
# 一般身份使用者，当然不能删除该项工作啦！
```

```
3. 删除在队列当中的打印工作
[root@linux ~]# lprm -P [printer 队列] job_id
参数：
-P  ：后面直接指定某部打印机的某个工作号码。注意，那个 job_id
      就是刚刚我们使用 lpq 查看到的那个 Job 的号码啦！

范例四：将刚刚的看到的那个 job 3 工作删除！
[root@linux ~]# lprm 3
[root@linux ~]# lpq -a
no entries
# 瞧！当然不见去啦！因为工作被我删除了嘛！

4. 以 cancel 删除在队列当中的打印工作
[root@linux ~]# cancel [-a] job_id
参数：
-a  ：不论队列里面有多少等待打印的工作，全部移除！
[root@linux ~]# cancel 3
[root@linux ~]# cancel -a
```

其实，整个 Linux 本机上面的打印信息，到这个时候就已经完成啦！ 不过，如果你还想要作额外的控制，或者是想要进行网络分享这部打印机时，嘿嘿！ 还得额外的加工，作一些手脚才行喔！底下就告诉你怎么作吧！

---

5. 权限控制（打印分享啦！）

我们前面提到的几个指令都可以直接在 CUPS 上面将打印的工作打印出来～ 不过，有的时候我们由于在进行打印机维护的作业，可能暂时不允许人家使用我们的打印机， 又不想让人家发现打印机目前有问题～怎么办？呵呵！这个就与 disable 这个指令有关啦！

```
1. 暂时关闭/启动打印机是否开始打印(但一定都可以接受工作到队列)
[root@linux ~]# /usr/bin/disable [-c] [printer 队列]
[root@linux ~]# /usr/bin/enable [printer 队列]
参数：
-c  ：将后面接的那个打印机队列所等待的工作，全部都删除。一般不会加上这个参数！
disable = cupsdisable
enable  = cupsenable
范例：

范例一：暂时取消刚刚建立的 laserjet2300 这部打印机的打印工作
[root@linux ~]# disable laserjet2300
[root@linux ~]# lpstat -t
```

```
scheduler is running
system default destination: laserjet2300
device for laserjet2300: ipp://192.168.10.119/
laserjet2300 accepting requests since Jan 01 00:00
printer laserjet2300 disabled since Jan 01 00:00 -
        Paused
# 发现否？目前的工作是暂停的！不过， scheduler 依旧是在进行！
# 所以，当然还可以继续接受队列的需求喔！让我们来测试看看！

[root@linux ~]# lpr /etc/passwd
[root@linux ~]# lpq
laserjet2300 is not ready
Rank    Owner   Job     File(s)                         Total Size
1st     root    5       passwd                          4096 bytes
# 嘿嘿嘿嘿！是 not ready ～还没有准备好嘛！
[root@linux ~]# cancel 5

范例二：开始重新让 laserjet2300 可以开始打印啊！
[root@linux ~]# /usr/bin/enable laserjet2300
[root@linux ~]# lpstat -t
scheduler is running
system default destination: laserjet2300
device for laserjet2300: ipp://192.168.10.119/
laserjet2300 accepting requests since Jan 01 00:00
printer laserjet2300 is idle.  enabled since Jan 01 00:00
# 注意一下，因为 bash 本身就含有一个 enable 的内部指令，
# 所以，建议您输入绝对路径来执行 enable ，或者直接改以 cupsenable 来执行！
```

这个 cupsdisable 与 cupsenable 可以支持让管理员拥有一段打印机维护的时间， 同时又还可以继续接受来自使用者的打印需求，真是不错啊！不过， 万一我就是明确的不要接受来自使用者的打印工作呢？也就是说，我连等待的工作都不开放， 只要有打印工作需求时，就回报『不能使用』。呵呵！就是 accept/reject 的工作啦！

```
2. 暂时停止/开启打印机队列的功能
[root@linux ~]# reject [printer 队列]
[root@linux ~]# accept [printer 队列]

范例一：暂时关闭 laserjet2300 吧！
[root@linux ~]# reject laserjet2300
[root@linux ~]# lpstat -t
scheduler is running
system default destination: laserjet2300
device for laserjet2300: ipp://192.168.10.119/
```

```
laserjet2300 not accepting requests since Jan 01 00:00 -
        Rejecting Jobs
printer laserjet2300 is idle.  enabled since Jan 01 00:00
        Rejecting Jobs

[root@linux ~]# lp /etc/passwd
lp: unable to print file: server-error-not-accepting-jobs
# 了解了吗？如此一来，连工作进入打印等待都不行喔！

范例二：启动 laserjet2300 吧！
[root@linux ~]# accept laserjet2300
```

很好玩吧！透过这四个小东西，我们就可以拥有一段管理维护打印机的时间啦～ ^_^

透过这些简单的指令，还有一些简单的编辑动作，您的打印机就可以在 Linux 上头顺利的运作了呢！ 而且还可以支持多部打印机同时存在，真是好方便啊！^_^

- 一个简单的练习

假设你目前的 FC4 主机上面接着一台 USB 接口的打印机， 你的系统并不允许使用 setup 的 Printer configuration 设定，这台 USB 接口的打印机是 Samsung 的 ML-1210 打印机，请问,您可以如何在 FC4 上面安装这部打印机？

- 先下载 PPD 定义档，档名为： Samsung-ML-1210-gdi.ppd 到 /usr/share/cups/model/ 当中；
- 加入打印机，使用下列方法：

```
[root@linux ~]# lpadmin -p samsung -v usb:/dev/usb/lp0 \
> -m Samsung-ML-1210-gdi.ppd -E
```

- 开始给他测试练习一下： 『 lp /etc/passwd 』如果有东西印出来，那就是 OK 啦！

Tips:
事实上，并不是所有的打印机厂商都有针对 Linux 操作系统释出相关的驱动程序， 所以，如果您想要购买能够让 Linux 使用的打印机时，还是建议您先前往上面提到的 http://www.linuxprinting.org/printer_list.cgi 这个网页进行查询～ 比如，HP 等大厂的打印机对于 Linux 的支持度就很不错～而如果您有旧的打印机， 例如鸟哥的 Lexmark X6150 这部机器，呵呵～那就不要想太多了～安装上面，实在很麻烦～ @_@ 但这并不是 Linux 的问题，您可以发起一人一信，寄信给打印机开发商，让他们重视 Linux 使用者群吧！



另外，如果老是看到屏幕前面显示：『 Printer not connected; will retry in 30 seconds...』， 很有可能是因为我们的装置代号输入错误，请使用『 lpstat -t 』查阅一下是否正确的设定好了？ 基本上，安装一部 Linux 有支持的打印机，真的是快速啦！

- 利用 web 接口管理 CUPS 打印机

除了使用手动的方式增加打印机之外，其实，我们还可以透过 web 接口来管理这部打印机喔！ 其实是管理 CUPS 这个 daemon 啦！如果您是在 Linux 本机前面使用 X Window 的话， 那么什么都不必想，直接在 X Window 当中的浏览器输入『 http://localhost:631 』 就可以进入管理画面了。如果像鸟哥一样，我的 Linux 主机都是不开 X Window 的， 那就得要作一些手脚啦！

首先，我们必须要让 CUPS 接受来自内部网段的 Client 端管理，假设我的内部网段是 192.168.1.0/24 ，那么我就得要这样做：

```
1. 修改 /etc/cups/cupsd.conf 的权限控制
[root@linux ~]# vi /etc/cups/cupsd.conf
# 找到底下的字眼，新增特殊字体的部分！
Port 631 <==确定一下，大约在 434 行左右，会有这个段落

<Location />          <==这个大约在 773 行左右
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From 192.168.1.0/24 <==加入这一行吧！
</Location>

<Location /admin>  <==这个大约是在 831 行左右。
AuthType Digest    <==把这里作个修改！
AuthClass System
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From 192.168.1.0/24  <==加入这一行吧！
</Location>

# 如果你曾经使用 seutp 内的 Printer configuration 设定过打印机的话，
# 那应该会看到底下这些字眼，注意，将这些资料通通删除！
<Location /printers/lexmark>
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
AuthType None
</Location>
# Lines below are automatically generated - DO NOT EDIT
Browsing On
```

```
BrowseProtocols cups
BrowseOrder Deny,Allow
BrowseAllow from @LOCAL
Listen 127.0.0.1:631
# 例如上面这 13 行字，全部通通删除吧！否则会造成问题！


2. 重新启动 cups 吧！
[root@linux ~]# /etc/init.d/cups restart

[root@linux ~]# netstat -tlnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address State  PID/Program name
tcp        0      0 0.0.0.0:631    0.0.0.0:*       LISTEN 28018/cupsd
# 注意啊！您开启的监听网域，必须要是 0.0.0.0 才对，如果出现 127.0.0.1 ，
# 那就表示您的 cupsd.conf 设定错误啦！

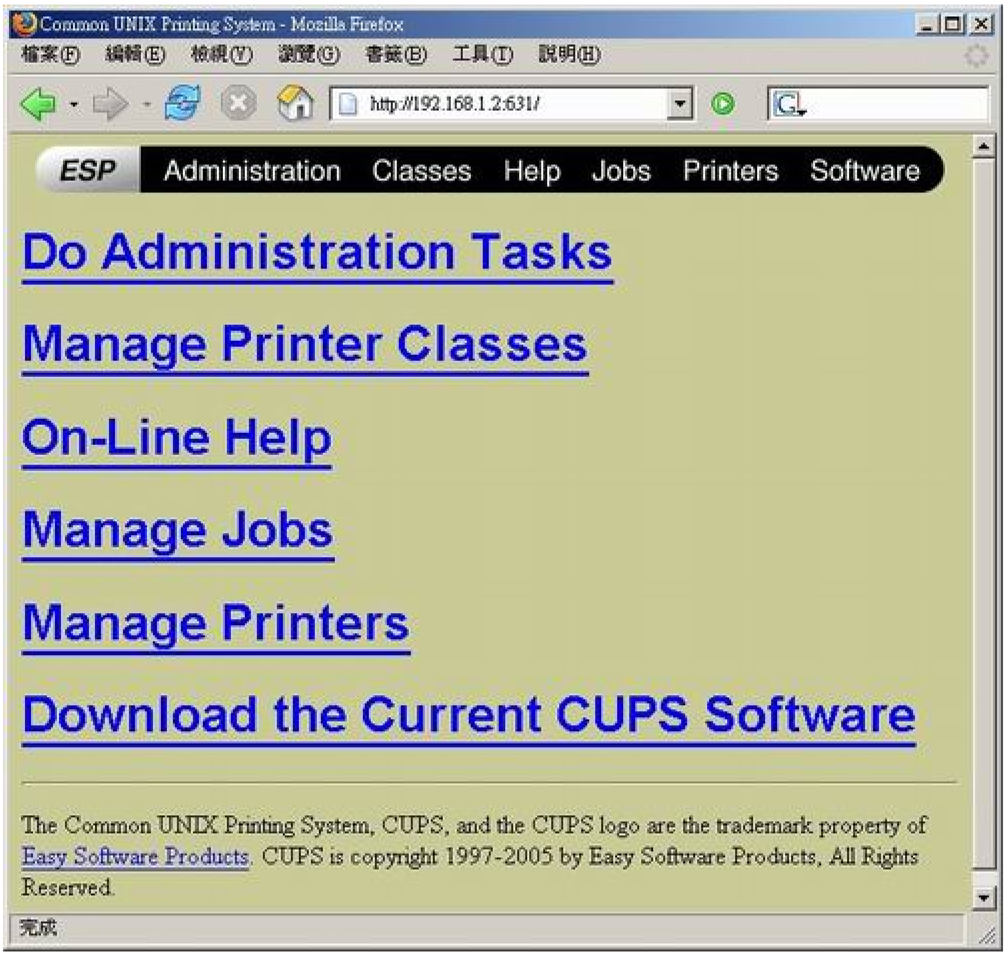

3. 设定管理打印机的管理员账号密码
[root@linux ~]# lppasswd [-a] [-x] [username]
参数：
-a  ：新增一个管理打印机的账号
-x  ：删除该账号
注意，该账号是我们可以随意取的，不一定要在 /etc/passwd 里面！

[root@linux ~]# lppasswd -a printermaster
Enter password:
Enter password again:

[root@linux ~]# cat /etc/cups/passwd.md5
printermaster:sys:a22ad518d345467ae72d3eb2cf4cdcc1
# 这就是我们利用 lppasswd 建立起来的密码数据啊！
```

接下来呢？呵呵！直接到区域内的任何一部计算机上面，启动浏览器，直接输入：

『 http://192.168.1.2:631 』假定我的 Linux 主机为 192.168.1.2 ，那就会看到如下画面：



图、利用 CUPS 的 Web 接口管理打印机

共有六大选项，我们比较经常使用的应该是 Printers 以及 Administration 这两项而已。 在按下了 Printer 项目之后，就会出现如下的画面：



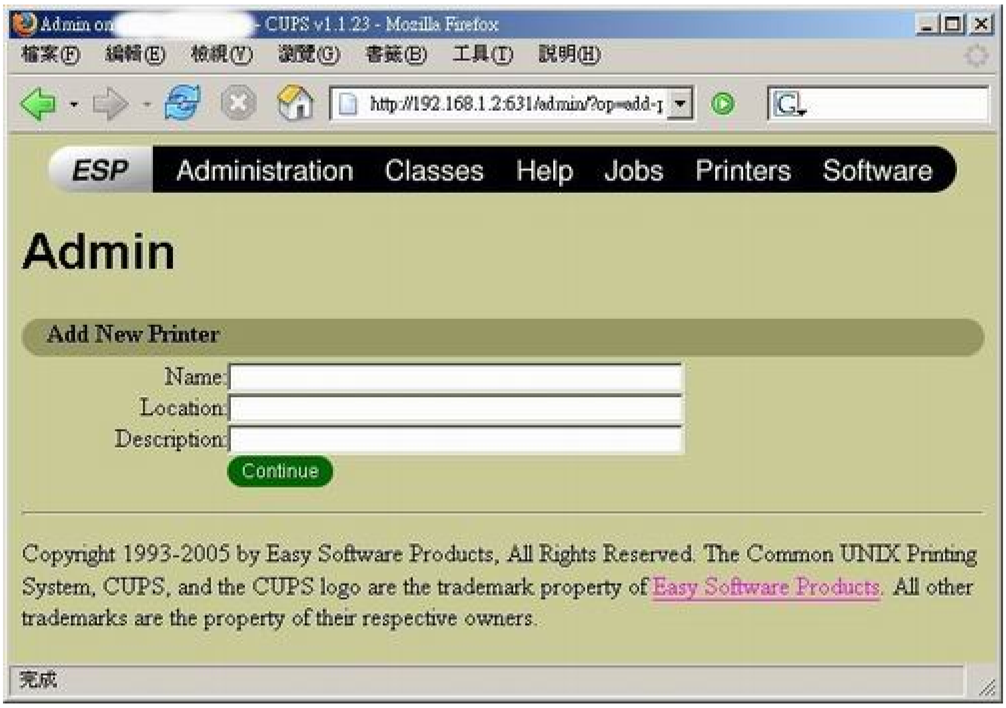
图、利用 CUPS 的 Web 接口管理打印机

看到了吗？刚刚我们设定成功的打印机已经在上头啦！ 而且还可以看到各部分详细的数据，很不错吧！如果想要进行一些额外的参数设定， 例如按下上图当中，左下角的 Add Printer ，就会出现如下的认证窗口：



图、利用 CUPS 的 Web 接口管理打印机

输入刚刚我们才以 lppasswd 建立的账号与密码，那么立刻就会出现如下的画面了：



图、利用 CUPS 的 Web 接口管理打印机

上图的画面其实与 setup 里面的新增打印机过程差不多， 只是上面的画面是以图形接口显示出来的而已啊！ ^_^。更多的使用方式， 就有待您自己去发掘啰！ ^_^

另外，如果你老是试不出来认证的话，那么干脆把 /etc/cups/cupsd.conf 里面的『AuthType 与 AuthClass』 这两个参数关掉，如此一来，就不会要求你输入账号密码，会直接让您进入打印机管理员的管理接口喔！

特殊的 filesystem： LVM

在第二篇的内容里面，我们主要谈到了磁盘档案系统，也晓得了，如果在安装初期， 没有做好整个硬盘的规划时，那么未来要新增磁盘空间，会很麻烦～～不过，这个问题在 LVM 面前， 似乎影响就不大了！为什么呢？因为 LVM 可以整合多个实体 partition 在一起， 让这些 partitions 看起来就像是一个磁盘一样！而且，还可以在未来新增其它的实体 partition 到这个 LVM 管理的磁盘当中。 如此一来，整个磁盘空间的使用上，实在是相当的具有弹性啊！

什么是 LVM： PV, PE, VG, LV

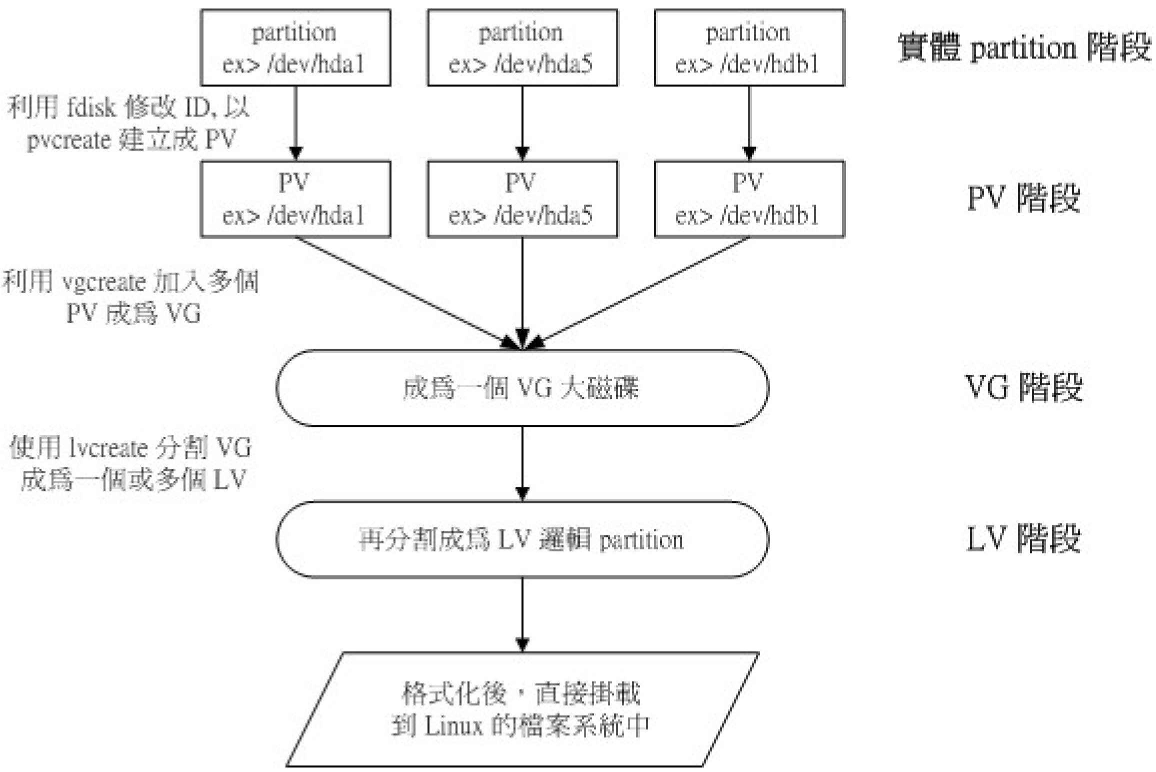
既然要玩 LVM 的话,那就得对 LVM 有点了解才行啊!事实上，LVM 其实就是将几个实体的 partitions 透过软件组合成一块看起来像是独立的大磁盘，而要用这块大磁盘，就得要再将他分割成为可以使用的 partition 才行!而我们知道每个 partition 上面的 filesystem 因为 block 大小的不同而有限制， 同样的， LVM 的大磁盘大小也是有限制的，主要是一个称为 PE 的咚咚。我们先来作一些简单的解释吧！

- Physical Volume, 简称 PV：
  这个就是实体磁盘啦！我们必须要将原本的磁盘，例如 /dev/hda5, /dev/hda6 等等的 partition ，利用 fdisk 等软件，将他们的 ID 改为 LVM(8e) ，并且修改磁盘的相关信息， 让

他成为 LVM 可以使用的磁盘才行。什么是 ID 啊？还记得使用 fdisk -l 看到的数据吧？ ID 83 是 Linux 的 partition ， 82 则是 Swap 的代号！这样瞭了吧？ 一块磁盘变成 PV 后， LVM 才能够利用该 partition 喔！重要重要！

- Volume Group, 简称 VG：
  其实我们 LVM 主要的目的就是要建立这个 VG 啦！他主要就是将刚刚的一个或多个 PV 组合成为一个大磁盘～这个大磁盘可以作为后续的分割之用喔！ 那么这个大磁盘的容量最大可到多大呢？ 最大容量的值与底下的 PE 有关， 如果完全使用 LVM 的预设参数时，那么一个最大的 LVM 磁盘可达到 256 GBytes。
- Physical Extend, 简称 PE：
  在建立 VG 的时候，我们同时需要指定 PE 这个数值！如果不指定的话，他预设是 4MB 的大小。 当 PE 为 4MB 时， VG 最大的容量就是 256 GBytes 啦！那么这个 PE 是什么玩意儿？？ 我们在 磁盘档案系统 那个章节当中提到的 inode, block 与 filesystem 大小的相关性当中， 有提到在 ext2/ext3 档案系统的格式化过程中，不同的 block 大小将会影响到整个 filesystem 大小的支持度。那这个 PE 其实就有点像是 VG 的 block 啦！ 所以他的大小将会影响到 VG 的最大值喔！如果你想要让你的 VG 大于预设的 256 GB 时， 记得要修改这个数值！(其实，一个 VG 最大可以容许 65534 个 PE ， 所以，修改 PE 值，当然就会影响到最大的 VG 容量啦！)
- Logical Volume, 简称 LV：
  这个 LV 就是最后被挂载到档案系统的 partition 啰～这个 LV 是由 VG 分割来的啦～ 他会建立一个装置代号，例如 /dev/vgname/lvname 在您的系统当中啊！

透过 PV, VG, LV 的规划之后，再利用 mkfs (mke2fs -j) 等等就可以将您的多个 partition 整合成为一个大磁盘， 再利用这个大磁盘来分割与格式化，就 OK 的啦！而且，这个大磁盘可以进行增加、减少容量的变化， 也就是说，这个 VG 大磁盘可以抽换 PV 哩！并且原有的数据，理论上，并不会被影响喔！是否很棒啊！ 整个 LVM 的处理流程与各组件之间的相关性，我们直接以下图来看看吧！



图、LVM 各组件之间的相关性

如此一来，我们就可以利用 LV 这个玩意儿来进行系统的挂载了。不过，您应该会觉得奇怪的是， 那么我的数据写入这个 LV 时，到底他是怎么写入硬盘当中的？ 呵呵！好问题～其实，依据写入机制的不同，而有两种方式：

- 线性模式 (linear)：假如我将 /dev/hda1, /dev/hdb1 这两个 partition 加入到 VG 当中，并且整个 VG 只有一个 LV 时，那么所谓的线性模式就是：当 /dev/hda1 的容量用完之后，/dev/hdb1 的硬盘才会被使用到。
- 交错模式 (triped)：那什么是交错模式？很简单啊，就是我将一笔数据拆成两部分， 分别写入 /dev/hda1 与 /dev/hdb1 的意思。如此一来，一份数据用两颗硬盘来写入， 理论上，读写的效能会比较好。

基本上，LVM 最主要的用处是在制造产生一个大磁盘， 而不是在建立一个效能为主的磁盘上，所以，我们应该利用的是 LVM 可以弹性管理整个 partition 大小的用途上，而不是着眼在效能上的。因此， LVM 预设的读写模式是线性模式啦！ 如果您使用 triped 模式，要注意，当任何一个 partition 『归天』时，所有的数据都会『损毁』的！ 所以啦，不是很适合使用这种模式啦！如果要强调效能与备份，那么就直接使用 RAID 即可， 不需要用到 LVM 啊！这样说，应该可以接受吧！

总之，鸟哥认为，整个 LVM 最大的用途即是在弹性管理磁盘的容量，让你的磁盘可以随时放大或缩小， 方便您将剩余的磁盘空间作一个较为良好的应用！

制作一个可使用的 LVM 磁盘流程

要让你的 Linux 使用 LVM 的功能，除了核心必须要有支持之外，你也必须要安装 lvm2 这个套件才行啊！好在，FC4 与其它较新的 distributions 目前的预设核心都有支持 LVM 的，这个就不需要担心了～呵呵。另外， lvm2 似乎也是预设安装的， 也不需要担心！真是好棒啊！ ^_^

整个 LVM 的制作流程，就跟我们上个小节提到的那张图一样，先制作 PV，再产生 VG， 最后分割出 LV 后，就可以格式化与挂载啰！OK！那我们底下就一步一步来实作看看吧！ 先说明一下鸟哥的环境，我的环境是 FC4 ，有一个 30 GB 的硬盘放在 /dev/hdb ， 这颗硬盘原本就已经被分割成为三个 partition ，分别为 /dev/hdb1, /dev/hdb2, /dev/hdb3， 各大约有 10GB 左右的容量。如果查阅一下系统，他会这样显示：

```
[root@linux ~]# fdisk -l /dev/hdb
Disk /dev/hdb: 30.7 GB, 30738677760 bytes
16 heads, 63 sectors/track, 59560 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1       19377     9765976+  83  Linux
/dev/hdb2           19378       38754     9766008   83  Linux
/dev/hdb3           38755       59560    10486224   83  Linux
```

现在，我想要制作一个 LVM 的磁盘出来，首先，我想要将 /dev/hdb1, /dev/hdb2 这两个 partition 加入这个 LVM 当中，来谈一谈怎么制作吧！ ^_^

- 建立 PV 与 PV 的查询：

要建立 PV 其实很简单，只要直接使用 pvcreate 即可！我们来谈一谈与 PV 有关的指令吧！

- pvcreate ：将实体 partition 建立成为 PV ；
- pvscan ：搜寻目前系统里面任何具有 PV 的磁盘；
- pvdisplay ：显示出目前系统上面的 PV 状态；
- pvremove ：将 PV 属性移除，让该 partition 不具有 PV 属性。
- partprobe ：这个指令可以让核心立刻读入最新的 partition table 而不必 reboot。

那就直接来瞧一瞧吧！

```
1. 先建立磁盘成为 ID 为 8e 吧！
[root@linux ~]# fdisk /dev/hdb
Command (m for help): p

Disk /dev/hdb: 30.7 GB, 30738677760 bytes
16 heads, 63 sectors/track, 59560 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1       19377     9765976+  83  Linux
/dev/hdb2           19378       38754     9766008   83  Linux
/dev/hdb3           38755       59560    10486224   83  Linux
```

```
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 8e
Changed system type of partition 1 to 8e (Linux LVM)

Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 8e
Changed system type of partition 2 to 8e (Linux LVM)

Command (m for help): p

Disk /dev/hdb: 30.7 GB, 30738677760 bytes
16 heads, 63 sectors/track, 59560 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1       19377     9765976+  8e  Linux LVM
/dev/hdb2           19378       38754     9766008   8e  Linux LVM
/dev/hdb3           38755       59560    10486224   83  Linux

Command (m for help): w
The partition table has been altered!
# 瞧到了吗？没错！整个 ID 被更改了！

[root@linux ~]# partprobe
# 这个指令有趣了！他可以让核心立刻读取最新的 partition table
# ，而不需要重新开机喔！



2. 开始将原本的磁盘制作成为 PV 的格式喔！
[root@linux ~]# pvscan
  No matching physical volumes found
[root@linux ~]# pvcreate /dev/hdb1
  Physical volume "/dev/hdb1" successfully created
[root@linux ~]# pvcreate /dev/hdb2
  Physical volume "/dev/hdb2" successfully created
[root@linux ~]# pvscan
  PV /dev/hdb1         lvm2 [9.31 GB]
  PV /dev/hdb2         lvm2 [9.31 GB]
  Total: 2 [18.63 GB] / in use: 0 [0   ] / in no VG: 2 [18.63 GB]
# 刚刚我们将整个 partition 改成为 PV 格式后，利用 pvscan
```

```
# 就可以看到整体的 PV 状态了。如果要看的更详细，那就如下所示：
[root@linux ~]# pvdisplay
  --- NEW Physical volume ---
  PV Name               /dev/hdb1
  VG Name
  PV Size               9.31 GB
  Allocatable           NO
  PE Size (KByte)       0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               CW7uAt-ZJa3-BMxe-tfti-16WU-OfNV-BQM6d8

  --- NEW Physical volume ---
  PV Name               /dev/hdb2
  VG Name
  PV Size               9.31 GB
  Allocatable           NO
  PE Size (KByte)       0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               1lc2SK-tfGN-ReOr-3mNi-nDAp-mpmb-1HUQFG
# 没错吧！确实建立起来了！基本上， PV 的名称与实际的 partition 刚好相同！
# 不过， VG 可就不是这么回事了。由于我们尚未对这两个 PV 分配 VG，
# 所以上面显示的 VG Name 就没有任何资料啊！
```

讲是很难，作是很简单！这样就将 PV 建立了两个啰！简单到不行吧！ ^_^

- 开始建立 VG ：

建立 VG 及 VG 相关的指令也不少，我们来看看：

- vgcreate ：就是主要建立 VG 的指令啦！他的参数比较多，等一下介绍。
- vgscan ：搜寻系统上面是否有 VG 存在？
- vgdisplay ：显示目前系统上面的 VG 状态；
- vgextend ：在 VG 内增加额外的 PV ；
- vgreduce ：在 VG 内移除 PV；
- vgchange ：设定 VG 是否启动 (active)；
- vgremove ：删除一个 VG 啊！

与 PV 不同的是， VG 的名称是自订的！我们知道 PV 的名称其实就是 partition 的装置代号， 但是这个 VG 名称，则是可以随便你自己取啊！在底下的例子当中， 我将 VG 名称取名为 vbirdvg ，所以建立这个 VG 的流程是这样的：

```
1. 先建立 VG 吧！
[root@linux ~]# vgcreate vbirdvg /dev/hdb1 /dev/hdb2
  Volume group "vbirdvg" successfully created
# 整个 vgcreate 的语法很简单，就是利用 vgcreate VGname PVname1 PVname2..
# 不过，如果想要修改前面提到的 PE 参数时，就得要加入 -s PE 数值了！

[root@linux ~]# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "vbirdvg" using metadata type lvm2

[root@linux ~]# vgdisplay
  --- Volume group ---
  VG Name               vbirdvg
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               18.62 GB
  PE Size               4.00 MB
  Total PE              4768
  Alloc PE / Size       0 / 0
  Free  PE / Size       4768 / 18.62 GB
  VG UUID               AZRSJx-FWYF-UIlH-NCh5-NqKS-f4gx-ZR049N


2. 尝试抽换一下 PV 吧！
[root@linux ~]# vgreduce vbirdvg /dev/hdb2
  Removed "/dev/hdb2" from volume group "vbirdvg"
[root@linux ~]# vgdisplay
  --- Volume group ---
  VG Name               vbirdvg
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  2
```

```
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               9.31 GB
  PE Size               4.00 MB
  Total PE              2384
  Alloc PE / Size       0 / 0
  Free  PE / Size       2384 / 9.31 GB
  VG UUID               AZRSJx-FWYF-UIlH-NCh5-NqKS-f4gx-ZR049N

[root@linux ~]# vgextend vbirdvg /dev/hdb2
  Volume group "vbirdvg" successfully extended
# 基本上，不难吧！这样就可以抽换整个 VG 的大小啊！
# 另外，可以使用 pvscan 看一下 PV 与 VG 的相关性喔！

[root@linux ~]# pvscan
  PV /dev/hdb1   VG vbirdvg   lvm2 [9.31 GB / 9.31 GB free]
  PV /dev/hdb2   VG vbirdvg   lvm2 [9.31 GB / 9.31 GB free]
  Total: 2 [18.62 GB] / in use: 2 [18.62 GB] / in no VG: 0 [0   ]
```

如此一来，我们就建立了一个好大好大的磁盘，他是整合了 /dev/hdb1 及 /dev/hdb2 啦！ 由于 VG 的最大容量与 PE 有关，PE 最多限制在 65534 个，但每个 PE 的大小则不固定。 预设 PE 是 4 MB ，如果你想要改成 16MB 的话，那就得要这样建立：

```
vgcreate -s 16M vbirdvg /dev/hdb1 /dev/hdb2
```

那么最大 VG 的容量就会由 256GB 增加为 1TB 啦！详细说明请参考 man vgcreate 喔！

- 建立 LV 吧！

创造出 VG 这个大磁盘之后，再来就是要建立分割区啦！ 这个分割区就是所谓的 LV 啰！假设我要将刚刚那个 vbirdvg 磁盘，分割成为 vbirdlv ， 整个 VG 都被分配到 vbirdlv 里面去！先来看看能使用的指令后， 就直接工作了先！

- lvcreate ：建立 LV 啦！
- lvscan ：查询系统上面的 LV ；
- lvdisplay ：显示系统上面的 LV 状态啊！
- lvextend ：在 LV 里面增加容量！
- lvreduce ：在 LV 里面减少容量；
- lvremove ：删除一个 LV ！
- lvresize ：对 LV 进行容量大小的调整！

```
1. 将整个 vbirdvg 通通分配给 vbirdlv 啊!
[root@linux ~]# lvcreate -L [sizeMG] -n [LVname] VGname
参数:
-L  : 后面接容量,容量的单位可以是 M, G 等等;
-n  : 后面接的就是 LV 的名称啦!
更多的说明应该可以自行查阅吧! man lvcreate

[root@linux ~]# lvcreate -L 18.62G -n vbirdlv vbirdvg
  Rounding up size to full physical extent 18.62 GB
  Logical volume "vbirdlv" created
[root@linux ~]# ll /dev/vbirdvg/vbirdlv
lrwxrwxrwx  1 root root 27 Nov 14 21:10 /dev/vbirdvg/vbirdlv ->
/dev/mapper/vbirdvg-vbirdlv
# 看见了没有啊! ? 这就是我们最重要的一个玩意儿了!
# 未来所有要挂载的数据,通通是透过这个装置的!

[root@linux ~]# lvscan
  ACTIVE            '/dev/vbirdvg/vbirdlv' [18.62 GB] inherit
[root@linux ~]# lvdisplay
  --- Logical volume ---
  LV Name                /dev/vbirdvg/vbirdlv
  VG Name                vbirdvg
  LV UUID                B6kSrg-9LMG-gqVy-jjz8-x0gM-ya9S-XLFcZN
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                18.62 GB
  Current LE             4767
  Segments               2
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:0
```

如此一来,整个 partition 也准备好啦!

- partition 的格式化与挂载!

这个部分鸟哥我就不再多加解释了! 直接来进行吧!

```
[root@linux ~]# mke2fs -j /dev/vbirdvg/vbirdlv
[root@linux ~]# mkdir /mnt/lvm
[root@linux ~]# mount -t ext3 /dev/vbirdvg/vbirdlv /mnt/lvm
[root@linux ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda2              9920624   3029712   6378844  33% /
```

```
/dev/hda1               101086     16524     79343  18% /boot
/dev/hda5             19236308    190776  18068380   2% /disk1
/dev/hda6            161836268     94272 153521136   1% /models
/dev/shm                192528         0    192528   0% /dev/shm
/dev/mapper/vbirdvg-vbirdlv
                      19219156    176288  18066588   1% /mnt/lvm
[root@linux ~]# fdisk -l /dev/hdb
Disk /dev/hdb: 30.7 GB, 30738677760 bytes
16 heads, 63 sectors/track, 59560 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1       19377     9765976+  8e  Linux LVM
/dev/hdb2           19378       38754     9766008   8e  Linux LVM
/dev/hdb3           38755       59560    10486224   83  Linux
```

知道那边不同了吗？没错！原始的 partition 还存在，但是我们却可以使用额外的 LVM 功能，将来自不同的 partition 的容量整个整合在一起，实在是给他相当的有趣啊！ ^_^

让原有的 LVM 磁盘加大的方法： resize2fs

好了，了解了如何制作 LVM 之后，接下来则是比较进阶的使用啦！我们知道 vbirdvg 已经有两个 PV， 但是实际上，我们还有个 /dev/hdb3 的实体 partition 啊！那么如何将这个 /dev/hdb3 加入到 vbirdvg ，且让 vbirdlv 增加呢？其实也不难啦！你必须要：

1. 将欲处理的 LV 卸载；
2. 建立 PV；
3. 将建立的 PV 以 vgextend 增加到 VG 当中；
4. 利用 lvextend 增加刚刚 VG 所增加的容量；
5. 利用 resize2fs 将 LV 的容量确实增加！

```
[root@linux ~]# umount /mnt/lvm

[root@linux ~]# fdisk /dev/hdb
Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): 8e
Changed system type of partition 3 to 8e (Linux LVM)

Command (m for help): w
The partition table has been altered!

[root@linux ~]# partprobe
[root@linux ~]# pvcreate /dev/hdb3
```

```
  Physical volume "/dev/hdb3" successfully created

[root@linux ~]# vgextend vbirdvg /dev/hdb3
  Volume group "vbirdvg" successfully extended
[root@linux ~]# vgdisplay
  --- Volume group ---
  VG Name               vbirdvg
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  5
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               28.62 GB
  PE Size               4.00 MB
  Total PE              7327
  Alloc PE / Size       4767 / 18.62 GB
  Free  PE / Size       2560 / 10.00 GB
  VG UUID               AZRSJx-FWYF-UIlH-NCh5-NqKS-f4gx-ZR049N
# 没错的！是增加了！也可以使用 pvscan 查阅喔！
# 而且剩余的没有使用的 VG 容量是 10.00 GB 喔！看清楚这个地方！ ^_^

[root@linux ~]# pvscan
  PV /dev/hdb1   VG vbirdvg   lvm2 [9.31 GB / 0     free]
  PV /dev/hdb2   VG vbirdvg   lvm2 [9.31 GB / 0     free]
  PV /dev/hdb3   VG vbirdvg   lvm2 [10.00 GB / 10.00 GB free]
  Total: 3 [28.62 GB] / in use: 3 [28.62 GB] / in no VG: 0 [0   ]

[root@linux ~]# lvextend -L +10G /dev/vbirdvg/vbirdlv
  Extending logical volume vbirdlv to 28.62 GB
  Logical volume vbirdlv successfully resized

[root@linux ~]# lvdisplay
  --- Logical volume ---
  LV Name                /dev/vbirdvg/vbirdlv
  VG Name                vbirdvg
  LV UUID                B6kSrg-9LMG-gqVy-jjz8-x0gM-ya9S-XLFcZN
```

```
  LV Write Access        read/write
  LV Status              available
  # open                 1
  LV Size                28.62 GB
  Current LE             7327
  Segments               3
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:0

[root@linux ~]# mount -t ext3 /dev/vbirdvg/vbirdlv /mnt/lvm
[root@linux ~]# df /mnt/lvm
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/vbirdvg-vbirdlv
                      19219156    176288  18066588   1% /mnt/lvm
```

看到了吧？？真是伤脑筋～怎么会 lv 已经增加了，但是 /dev/vbirdvg/vbirdlv 却没有加大呢？ 这是因为我们的 ext3 主要信息在最初规划时，就已经写入 super block ， 同时 inode 与 block 数量又是固定的，所以，这个容量大小当然不会有变化！那怎么办？ 没关系，我们可以使用 ext2/ext3 的工具程序来处理！

```
[root@linux ~]# resize2fs [-f] [device] [size]
参数：
-f      ：强制进行 resize 的动作！
[device]：装置代号啊！
[size]  ：可以加也可以不加。如果加上 size 的话，那么就必须要给予一个单位，
          譬如 M, G 等等。如果没有 size 的话，那么预设使用『整个 partition』
          的容量来处理！

[root@linux ~]# umount /mnt/lvm
[root@linux ~]# resize2fs -f /dev/vbirdvg/vbirdlv
resize2fs 1.38 (30-Jun-2005)
Resizing the filesystem on /dev/vbirdvg/vbirdlv to 7502848 (4k) blocks.
The filesystem on /dev/vbirdvg/vbirdlv is now 7502848 blocks long.

[root@linux ~]# mount -t ext3 /dev/vbirdvg/vbirdlv /mnt/lvm
[root@linux ~]# df /mnt/lvm
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/vbirdvg-vbirdlv
                      29540436    176288  28163696   1% /mnt/lvm
```

嘿嘿！真的放大了吧！！而且如果你已经有填数据在 LVM 扇区当中的话！ 这个数据是不会死掉的喔！还是继续存在原本的扇区当中啦！ 整个动作竟然这么简单就完成了！原本的数据还是一直存在而不会消失～您说， LVM 好不好用啊！

事实上， resize2fs 也可以用来处理 ext2/ext3 原本 filesystem 的大小。 可以先以 fdisk 处理分割区，再以这个 resize2fs 来放大或者缩小 partition 啊！ 要注意，待处理的 partition 不可挂载喔！

注意事项：

先来说一说，你必须要知道的一些 LVM 的指令整理：

任务	PV	VG	LV
搜寻(scan)	pvscan	vgscan	lvscan
建立(create)	pvcreate	vgcreate	lvcreate
列出(display)	pvdisplay	vgdisplay	lvdisplay
增加(extend)		vgextend	lvextend
减少(reduce)		vgreduce	lvreduce
删除(remove)	pvremove	vgremove	lvremove
改变容量(resize)			lvresize

另外，还需要以 resize2fs 来修订档案系统实际的大小才行啊！ ^_^ 。至于虽然 LVM 可以弹性的管理你的磁盘容量，但是要注意， 如果你想要使用 LVM 管理您的硬盘时，那么在安装的时候就得要做好 LVM 的规划了， 否则未来还是需要先以传统的磁盘增加方式来增加后，移动数据后，才能够进行 LVM 的使用啊！

无论如何还是要跟大家报告的，鸟哥个人认为 LVM 比较适合用在制作一个具有弹性的磁盘容量的环境，并不是要制作一个高效能的储存设备的环境，所以鸟哥在 lvcreate 时， 才没有讲到如何制作 triped 模式的参数啊！如果真的想要制作一个高可靠性、 高性能的储存环境，那还是得重硬件来着手，譬如底下我们要谈一谈的 iSCSI 的 RAID 储存架构，应该就是个不错的选择喔！ ^_^

另外，如果你想要将 LVM 移除的话，那么就应该要：

1. 先卸载系统上面的 LVM 扇区；
2. 使用 lvremove 移除 LV ；
3. 使用 vgchange -a n VGname 让 VGname 这个 VG 不具有 Active 的标志；
4. 使用 vgremove 移除 VG：
5. 使用 pvremove 移除 PV；
6. 最后，使用 fdisk 修改 ID 回来啊！

好吧！那就实际的将鸟哥刚刚建立的 LVM 给他拿掉吧！

```
[root@linux ~]# umount /mnt/lvm
[root@linux ~]# lvremove /dev/vbirdvg/vbirdlv
Do you really want to remove active logical volume "vbirdlv"? [y/n]: y
  Logical volume "vbirdlv" successfully removed
[root@linux ~]# vgchange -a n vbirdvg
  0 logical volume(s) in volume group "vbirdvg" now active
```

```
[root@linux ~]# vgremove vbirdvg
  Volume group "vbirdvg" successfully removed
[root@linux ~]# pvremove /dev/hdb1
  Labels on physical volume "/dev/hdb1" successfully wiped
[root@linux ~]# pvremove /dev/hdb2
  Labels on physical volume "/dev/hdb2" successfully wiped
[root@linux ~]# pvremove /dev/hdb3
  Labels on physical volume "/dev/hdb3" successfully wiped
```

最后在用 fdisk 将磁盘的 ID 给他改回来 82 就好啦！整个过程就这样的啦！ ^_^

额外的储存设备 iSCSI 协议的磁盘阵列：

常常会听到所谓的 SAN (Storage Area Networks) 与 NAS (Network Attached Storage) 这两个字眼，NAS 主要是以一部专门给储存数据用的主机，以现有的 TCP/IP 协议来提供作为类似 file server 的用途， 他可以直接放置在网络上面，提供使用者存取数据之用。SAN 则类似一部储存设备， 藉由光纤信道提供某几部主机进行数据的存取之用，价格昂贵，维护成本高，但是储存效能佳！ 但不论是哪一种架构，基本上，他们的储存硬盘通常是以磁盘阵列 (RAID) 作为基础的。 底下我们会来谈一谈什么是磁盘阵列，并同时以目前较新的 iSCSI 架构的储存设备来进行一个简单的网络储存设备架设。

什么是磁盘阵列

磁盘阵列全名是『 Redundant Arrays of Inexpensive Disks 』，简写为 RAID， 他是透过一个技术(软件或硬件)，将多个较小的磁盘整合成为一个较大的磁盘装置； 而这个较大的磁盘功能可不止是储存而已，他还保有数据保护的功能呢。 整个 RAID 由于选择的等级 (level) 不同，而使得整合后的磁盘具有不同的功能， 基本常见的 level 有这几种：

- Linear mode (线性模式)：

两个以上的磁盘整合成为一个实体的储存装置；这个模式的特色是， 所有的数据是『一个一个填满后，才将数据继续写到下一个磁盘上』； 由于数据是一个一个写入到不同的硬盘当中，因此，整个磁盘的读取效能并不会增加， 此外，由于数据具有连续性，因此，若不小心有任何一个磁盘损毁时，嘿嘿！ 您的数据可能通通救不回来了～这种模式唯一的好处， 就是磁盘的空间完整的被利用完毕！不会有任何保留空间 (redundant)。

- RAID-0 (交错模式, stripe)：

这种模式主要是利用容量相同的磁盘来达成，效能会比较好。 所谓的交错 (stripe) 是因为档案数据是同步洒到不同的磁盘上头去的意思， 也就是说，假设我有两颗磁盘设定成 RAID-0 ，那么当我有 100MB 数据要写入时， 则 100MB 会被拆成两个 50MB 分别写入不同的磁盘上头去！

所以啰，因为每一个磁盘写入的数据量只有一半，因此，读写的效能都会大大的增加！ 而且越多颗磁盘所造成的 RAID-0 装置，理论上，效能增加的越明显。 但是这种模式有个最大的问题，那就是，因为一笔数据被拆成几个部分分布在不同的磁盘上头， 因此『任何一颗磁盘的损毁，都会让你的数据救不回来！』

另外，如果使用不同容量的磁盘来达成 RAID-0 时，则在储存数据长大到一定程度时， RAID-0 的效能会

变差。假设我用了一颗 20GB 两颗 16GB 的硬盘好了， 那么当总数据量少于 48GB (16x3) 时，效能是很不错的。 但是当超过 48GB 时，则数据仅能储存在 20GB 那一颗了 (可用容量剩下 4GB 啊！)， 所以啰，当然效能就变差了啊！

- RAID-1 (映像模式, mirror)：

这种模式也是需要相同的磁盘容量的，最好是一模一样的磁盘啦！如果是不同容量的磁盘组成 RAID-1 时，那么总容量将以最小的那一颗磁盘为主！ 这种模式主要是『让同一份数据，完整的保存在两颗磁盘上头』，也就是说，如果我有一个 100MB 的档案，且我仅有两颗磁盘组成 RAID-1 时， 那么这两颗磁盘将会同步写入 100MB 到他们的储存空间去， 因此，整体容量几乎少了 50%。由于两颗硬盘内容一模一样，好像镜子映照出来一样， 所以我们也称他为 mirror 模式啰～

我们可以说，这种模式最大的优点大概就是在进行备份吧！ 因为所有的数据都被存放在两个磁盘上面，所以，任何一颗磁盘损毁时，嘿嘿！ 所有的数据可都还是保存的好好的呢～

至于效能上面，由于要写入的数据变多了，(同步写入两颗硬盘嘛！)， 所以，效能会比单颗磁盘还要差一些，没办法，因为我们都是透过同一个总线在进行数据的通行啊～ 不过，读取的效能还不错，因为数据有两份，如果多个 processes 在读取同一笔数据时， RAID 会自行取得最佳的读取平衡。

事实上，为了保有 RAID-1 的储存优点，又想要具有类似 RAID-0 的效能增强，所以，后来也有所谓的 RAID 0+1 ，亦即同时具有 RAID-0 与 RAID-1 的功能； 只是，这样的功能至少需要四颗以上的相同容量的磁盘才行～ 而且总可用容量会减少一半 (因为 RAID-1 啊！)。

- RAID-5：

这个类型最有趣～也是目前最常见的一种类型了。 RAID-5 会整合多个磁盘 (通常需要三个以上)，然后每部磁盘驱动器上面都会记录少许的其它部磁盘驱动器的信息 (parity information)，由于这个动作，因此，实际上可以使用的容量其实是 (N-1)*S ，那个 N 为全部的磁盘驱动器总数，S 则是最小的那个磁盘的容量。 少掉的空间就是用来作为信息记录用的。

由于这个机制的存在，因此，当那 N 部磁盘驱动器有任何一部出问题时， 他的数据都会被平均记录到其它 N-1 部磁盘驱动器内，所以，只要你将坏掉的那一颗拿掉， 换一颗好的磁盘后，该磁盘原本的内容就会被重建 (rebuild) 起来， 呵呵～很棒的备份效果吧！

除此之外，因应目前所谓的热拔插 RAID 架构，因此，很多的硬件 RAID 在制作这个 RAID-5 的类型时，他会使用 N+1 颗磁盘，其中 N 颗用来作为 RAID-5 之用， 另外那一颗则做为磁盘损坏时的实时处理之用 (spare disk)。 所以，假设您有十颗磁盘在这样的架构下时，则事实上只有九颗在进行 RAID-5 的存取，一颗作为错误处理，所以总容量应该是 8xS 才对喔！而当有任何一颗磁盘出问题时， 那一颗 spare disk 就派上用场了，他会立刻被重建， 因此，您只要将有问题的磁盘拔掉，换上一颗新的，嘿嘿！搞定！

不过还是要注意啦，因为 RAID-5 仅能处理一颗硬盘坏掉时的处置，若同时有两颗以上的磁盘损毁，那....所有的数据还是会完蛋的！ 所以啦，特重要的信息还是得要复制出来才行～关于考虑，呵呵～请参考备份策略吧！

至于存取效能上面，读取的效能几乎可以媲美 RAID-0 ，但是写入的效能就无法像 RAID-0 那样明显。因为还得要计算分别写入到所有磁盘当中的平衡信息 (parity information)， 所以写入的效能虽然是会增加不少，不过，不容易计算出来实际的增加的效能啦～

呵呵！经过上面的说明，您应该会晓得，为何企业会需要使用磁盘阵列了吧？ 没错，因为 RAID 不但可以增加资料存取的效能， 而且对于备份与数据的可靠性而言，他具有相当程度的类似备份的功效，因此， 很适合需要大量存取数据的主机系统。所以啰，目前很多的在线储存设备， 基本上，都是透过 RAID 装置来达成的～底下我们就大略的来介绍一下目前挺流行的 iSCSI 接口的储存设备吧！(如果对于 Linux 上面的 RAID 有兴趣， 可以参考这一篇：http://www.tldp.org/HOWTO/ Software-RAID-HOWTO-1.html)

---

iSCSI 磁盘阵列的架设与使用

由于企业的数据量越来越大，而且重要性与保密性越来越高，尤其类似数据库的内容， 哇！常常容量单位是以 TB (1TB = 1024GB) 在进行计算的；可怕吧！ 所以啰，上一个小节内提到的磁盘阵列 (RAID) 的应用就很重要了。 不过， RAID 毕竟只是在一部主机上面的储存装置，如果想要提供给网络上面的其它 client 端来使用，可能还需要主机提供相关的服务才行啊！而且，透过网络主机来连接， 效能上可能是卡在网络传输速度，而不是 RAID 的速度说。而且， RAID 装在一部主机上面时， 能够提供给 client 端使用的情况有限啦～大多仅是数据而已。

- NAS

为了解决网络应用上面的，很多厂商提供了一些不错的想法。首先，就是那个 NAS (Network Attach Storage)，基本上， NAS 其实就是一部客制化好的主机了， 只要将 NAS 连接上网络，那么在网络上面的其它主机就能够存取 NAS 上头的数据了。 简单的说，NAS 就是一部 file server (档案服务器) 啰～ 不过， NAS 由于也是接在网络上面，所以，如果网络上有某个 client 大量存取 NAS 上头的数据时， 是很容易造成网络停顿的问题的，这个比较麻烦点～

> Tips:
> 在鸟哥的理解当中， NAS 基本上就是一部完整的主机， 他有独立的操作系统与运算、储存等处理单元，其它的 client 端只要能够与 NAS 的协议兼容，那么他就能够存取 NAS 上头的数据啊！



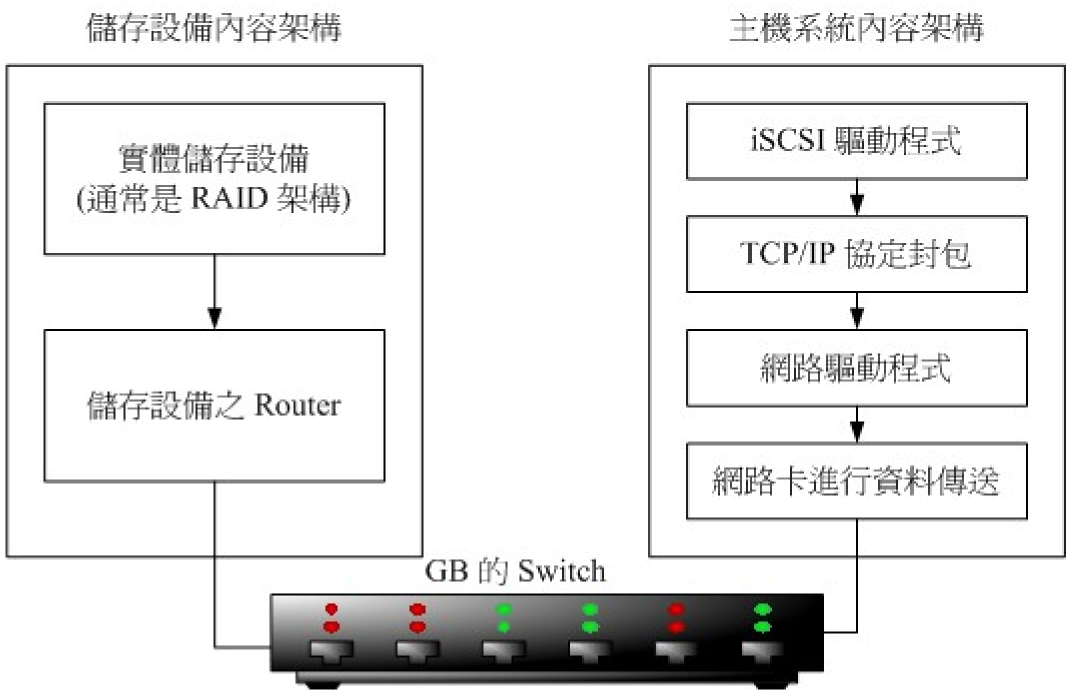
- SAN

我们需要大量的存取装置，目的就是为了要强化存取效能与总可用容量，现在知道总可用容量可以使用 RAID 技术来加强，但是存取效能呢？透过网络来存取时，确实可能导致存取效能的低落啊。 为了解决这个问题，因此后来很多厂商开发出所谓的光纤信道。光纤信道的速度要快上很多， 目前标准的光纤信道是 2GB ，未来还可能到达 10GB 以上呢～ 不过，使用光纤等技术较高的设备，当然就比较贵一些。 利用光纤，配合一些硬件的标准化，后来一些厂商提出了 SAN (Storage Area Network) 架构； SAN 利用较高阶的光纤信道来进行主机与储存设备的连接， 让主机透过光纤来快速存取储存设备内的数据，而不是透过较慢的网络架构。 但是 SAN 架构的初设成本太贵，而且维护费用颇高！

拜以太网络盛行，加上技术成熟之赐，现今的以太网络媒体 (网络卡、交换器、路由器等等设备) 已经可以达到 GB 的速度了，离 SAN 的光纤信道速度其实差异已经缩小很多啦～ 那么是否我们可以透过这个 GB 的以太网络来达到类似 SAN 的功效呢？ 没错！这就是我们接下来要提到的 iSCSI 架构啦！ ^_^

- iSCSI

其实，整个 iSCSI 的架构主要分为储存设备与提供 iSCSI 联机的主机端， 而 iSCSI 则透过以太网络连结储存设备与主机就是了。 两者的关系有点像底下这个图示：



图一、 iSCSI 的主机与储存设备的关系。

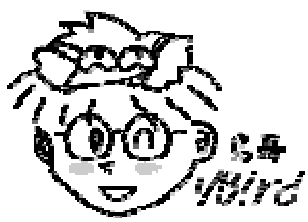
首先，您必须要拥有 iSCSI 接口的储存设备（废话！不然怎么玩？）， 然后，您的 iSCSI 储存设备上面必须要具有 IP 地址，并且储存设备以网络线连结到拥有 GB 速度的 Switch 上。再来，您的主机必须要启动 iSCSI 的驱动程序， 驱动他之后，他会使用 TCP/IP 的网络协议连结到储存设备上头， 并且『将整个储存设备视为主机上面的一个 SCSI 硬盘』的模样！ 也就是说，基本上，iSCSI 的储存设备就是您主机上面的一个实体硬盘， 只是这个硬盘是以 iSCSI 协议及 TCP/IP 协议来达成的。而且，他在主机上面的使用， 几乎就与实体 SCSI 硬盘没有两样！磁盘代号则为 /dev/sda... 。

经过上面的说明，你可以了解到，其实整个 iSCSI 的储存设备并不是一个完整的主机喔，他仅是一个储存设备， 跟上头提到的 NAS 并不相同，反而有点类似 SAN 啦～而且，这个储存设备可以让主机完整掌控，几乎就与主机上头的 SCSI 硬盘没有两样就是了。

底下鸟哥以 Promise 公司出品的 iSCSI 装置来进行说明， 该 iSCSI 装置里面已经存在 10 颗 250GB 的 SATA 硬盘，其中有一颗做为备份 (spare disk) ， 所以共有九颗硬盘作为 RAID-5 的架构。因此，整个 RAID 的容量则约略是： 250*(9-1)=2TB，不过，这样只是约略估计啦，因为 250GB 硬盘的计算是以 10 进位计算的， 跟我们常见的磁盘容量计算单位不相同。不过，很接近就是了！ ^_^

再来，由于 Promise 公司并没有提供 Linux 主机的 iSCSI 驱动程序，所以， 鸟哥则以 Linux-iSCSI project 提供的相关信息，并以底下这一篇为基底： http://www.cuddletech.com/articles/iscsi/index.html 安装鸟哥的 Red Hat 9 Linux 的 iSCSI driver 。基本上，在该篇文章中， Linux 主机被称为是 initiator 的啦！

Tips:
事实上，由于 iSCSI 储存设备与 Linux 主机之间，是透过 TCP/IP 协议， 亦即网络来进行数据的存放，因此，您至少必须要具备基础的网络基础知识才行～ 不过，这一篇实在挺有趣的，而单独写一篇服务器，似乎也没有那个必要， 所以，就让鸟哥放到这个地



方来了～所以啰，如果您目前看不懂也没关系啦； 而且，鸟哥也是刚好有机会接触到 iSCSI 的设备才能够玩这一篇啊!呵呵～ 其实还挺高兴的！ ^_^

基本上，鸟哥的环境架构是这样的：

- Linux 主机：
    - 系统： Red Hat 9
    - 核心： Red Hat 9 的预设核心 (2.4.20-8mp)
    - 软件： 已安装 kernel-source, make, gcc 等必要的套件；
    - iSCSI driver：http://sourceforge.net/project/showfiles.php?group_id=26396
    - IP：我的 Red Hat 9 IP 为 192.168.10.30

- iSCSI 储存设备架构：
    - 型号为 PROMISE M500i ，使用 iSCSI 机型；
    - 使用 10 颗硬盘，1 颗为 spare disk，其余 9 颗做成 RAID-5；
    - IP：iSCSI 储存设备的 IP 为 192.168.10.200
    - 连接到储存设备的账号与密码分别为： account/iscsipw

相关的 iSCSI 装置、网络接线的连结，以及 GB switch 的选购与连结等等， 请与相关的硬件厂商联系，他们会帮忙搞定的～我们要作的，仅是在 Linux 主机上面安装驱动程序， 并且将他挂载起来就是了。其实，当 iSCSI RAID 到您府上时，安装好之后， 我们就不需要动他的设定了～因为他仅提供储存空间嘛！只要控制 Linux 主机端即可啦！ 整个安装的步骤有点像这样：

---

1. 下载适当的驱动程序：

   由于鸟哥的测试机使用的是 Red Hat 9 ，他的核心是 2.4.xx 版本， 所以，我下载的是 3.6.3 的版本，当然，你也可以直接下载给 Red Hat 9 使用的 RPM 档案啊！：

    - iSCSI driver：http://sourceforge.net/project/showfiles.php?group_id=26396

---

2. 开始安装 iSCSI 驱动程序：

   iSCSI 的驱动程序安装真的很简单啊!我们刚刚下载的档案放置到 /usr/local/src 后， 然后直接解压缩，之后将他直接进行 make 与 make install 即可！ 鸟哥这里讲得很简单，若有需要更详细的 tarball 安装说明，请参考 原始码与 tarball 那个章节啊！ 此外，由于 iSCSI 是一种驱动程序，因此，他会读取 Linux kernel 的相关原始码档案， 所以，你也必须要确认你的系统上面确实含有 Linux kernel 在 /usr/src/linux 目录下才行！ 这也是很重要的一项准备工作喔！

```
1. 先进行解压缩的动作：
[root@linux ~]# cd /usr/local/src
[root@linux src]# tar -zxvf linux-iscsi-3.6.3.tgz
[root@linux src]# cd linux-iscsi-3.6.3c


2. 开始进行编译与安装
[root@linux linux-iscsi-3.6.3]# make clean && make
[root@linux linux-iscsi-3.6.3]# make install
# 首先，会有一些关于核心方面的相关说明，这里看看即可！
Note: using kernel source from /lib/modules/2.4.20-8smp/build containing
kernel version 2.4.20-8custom
Note: using kernel config from /boot/config-2.4.20-8smp
Installing iSCSI driver for Linux 2.4.20-8smp

# 再来，则会安装启动 iSCSI 的 script ，预设放置到 /etc/rc.d/init.d/iscsi 去！
# 同时注意一下，这个程序会自动的加入到 chkconfig 的管理项目当中，因此，
# 安装好的同时，这个 iscsi 就会在开机时主动的启动了。
The initialization script has been installed as /etc/rc.d/init.d/iscsi.
iSCSI has been set up to run automatically when you reboot.

# 同时，会将我们的 Linux 主机仿真成为 CISCO 的 iSCSI
# 协议的接受器，相关的设定数据会被写入到 /etc/initiatorname.iscsi 当中。
InitiatorName iqn.1987-05.com.cisco:01.d1dbb1112d38 has already been
generated and written to /etc/initiatorname.iscsi.

# 接下来这个档案才是我们所关心的！那就是连结到 iSCSI 储存装置的设定文件！
# 你必须要根据你的装置来给予相关的修改后，他才会顺利工作喔！
Make sure you check and edit the /etc/iscsi.conf file!


3. 开始进行修改的工作：
[root@linux ~]# vi /etc/iscsi.conf
# 在这个档案当中新增底下这几行，注意，账号、密码与 IP 均需确定正确喔！
Username=account
Password=iscsipw
DiscoveryAddress=192.168.10.200
  Username=account
  Password=iscsipw


4. 开始启动 iscsi 啰！
[root@linux ~]# /etc/init.d/iscsi start
```

```
Starting iSCSI: iscsi iscsid fsck/mount

[root@linux ~]# vi /var/log/messages
iscsid[3208]: version 3.6.3.0 variant (27-Jun-2005)
iscsid[3208]: root development build created Mon Jun 27 14:34:43 CDT 2005
iscsid[3209]: INBP boot check returned this_is_inbp_boot = 0
iscsid[3212]: Connected to Discovery Address 192.168.10.200
kernel: iSCSI: bus 0 target 0 = iqn.1994-12.com.promise.target.3b.31.
          4.55.1.0.0.20
kernel: iSCSI: bus 0 target 0 portal 0 = address 192.168.10.200 port
          3260 group 2
kernel: iSCSI: bus 0 target 0 trying to establish session f5ad6000 to
           portal 0, address 192.168.10.200 port 3260 group 2
kernel: iSCSI: bus 0 target 0 established session f5ad6000 #1 to portal
          0, address 192.168.10.200 port 3260 group 2, alias VTrak M500i
kernel: scsi singledevice 0 0 0 0
kernel:   Vendor: Promise   Model: VTrak M500i       Rev: 1122
kernel:   Type:   Direct-Access                      ANSI SCSI revision: 04
kernel: Attached scsi disk sda at scsi0, channel 0, id 0, lun 0
kernel: SCSI device sda: 3890624512 512-byte hdwr sectors (1992000 MB)
kernel:  sda:
# 看到了吗？没错！确实有连接到正确的储存位置了～
# 而且捉到的是 sda 这个装置代号喔！容量可有 1992000MB 这么大！
# 然后我们来看看系统是否能够捉到这个装置呢？

[root@linux ~]# fdisk -l
Disk /dev/sda: 1991.9 GB, 1991999750144 bytes
64 heads, 32 sectors/track, 1899719 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes

   Device Boot    Start       End    Blocks   Id  System

[root@linux ~]# iscsi-ls
*******************************************************************************
        SFNet iSCSI Driver Version ... 3.6.3 (27-Jun-2005 )
*******************************************************************************
TARGET NAME             : iqn.1994-12.com.promise.target.3b.31.4.55.1.0.0.20
TARGET ALIAS            : VTrak M500i
HOST NO                 : 0
BUS NO                  : 0
TARGET ID               : 0
TARGET ADDRESS          : 192.168.10.200:3260
SESSION STATUS          : ESTABLISHED AT Thu Nov 10 20:13:43 2005
```

```
NO. OF PORTALS          : 1
PORTAL ADDRESS 1        : 192.168.10.200:3260,2
SESSION ID              : ISID 00023d000001 TSIH 04
*******************************************************************************
# 看到了吧？使用 fdisk -l 可以看到名称为 /dev/sda 的装置，使用 iSCSI
# 提供的 iscsi-ls 也可以查询的到相关的信息呢！真是很不错啊！


5. 分割 /dev/sda 与格式化！
[root@linux ~]# fdisk /dev/sda
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1899719, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1899719, default 1899719):
Using default value 1899719
Command (m for help): w

[root@linux ~]# mke2fs -j /dev/sda1
```

经过这个简单的动作，我的 Linux 主机已经捉到 iSCSI 储存装置， 并且，他就好像是我自己 Linux 本机上面的一个 SCSI 硬盘一样！ 使用的方式几乎一模一样～没有差异呢！真是太方便了～接下来，当然就是要挂载啦！

---

3. 设定挂载：

假设我的这部 iSCSI 主机要挂载到 /cluster/raid 这个目录下，那么我就这样做：

```
[root@linux ~]# mkdir /cluster/raid
[root@linux ~]# mount -t ext3 /dev/sda1 /cluster/raid
[root@linux ~]# df
Filesystem          1K-blocks      Used Available  Use% Mounted on
/dev/hda1            10080488   2950408   6618012   31% /
/dev/hda2             5036316     81172   4699312    2% /var
/dev/hda3             3020172     33492   2833260    2% /tmp
none                   773736         0    773736    0% /dev/shm
/dev/sda1          1914788196  27040372 1790482212  2% /cluster/raid
```

很有趣吧！这样就能够将 iSCSI 挂载到我们的机器上啰～

---

4. 设定自动挂载：

   不过，如果你想要让系统自动挂载 iSCSI 装置的数据，该写入哪里？ 当然不可能是 /etc/fstab 啰～因为在加载 /etc/fstab 之前，根本就还没有驱动网络， 所以，当然不可能写入 /etc/fstab 。好在，我们的 iSCSI 有提供不错的模块， 你可以直接修改 /etc/fstab.iscsi 这个档案，将他设定成为：

```
[root@linux ~]# vi /etc/fstab.iscsi
/dev/sda1    /cluster/raid    ext3    defaults     0   0
```

   如此一来，开机就会自动的挂载该 iSCSI 的装置啦！就是这么简单啊！^_^

鸟哥是由于研究室有一部 iSCSI 的 Raid 才有可能接触到这么高档的货色～ 真是觉得很开心！整个 iSCSI 的设定并不困难，当然，要达成更高效率的读写数据， 可能还是需要进一步的研究啦！提供给您作为一个设定的参考啰！ ^_^

参考文献

- USB 的在线书籍：http://www.linux-usb.org/USB-guide/book1.html
- Linux USB ：http://www.linux-usb.org/
- LM_sensors 官方网站：http://www2.lm-sensors.nu/~lm78/
- Linux 磁盘阵列：http://www.tldp.org/HOWTO/Software-RAID-HOWTO-1.html
- Linux iSCSI project：http://linux-iscsi.sourceforge.net/
- A Quick Guide to iSCSI on Linux：http://www.cuddletech.com/articles/iscsi/index.html
- 工研院 iSCSI 简介：http://www.itri.org.tw/chi/southern_branch/ccl_01c.jsp
- 季明, "SuSE Linux Enterprise Server 9 管理手册", 第六章, 碁峰出版社。
- 日京三子的 LVM 笔记：http://phorum.study-area.org/viewtopic.php?t=19073

习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 如何建立 /dev/usb/lp8?

  首先，必须要查阅得该装置的主要装置代号，亦即 180，至于次要代号则是 8 ， 再使用 mknod 来建立，因此，需要这样做：
  mknod /dev/usb/lp8 c 180 8

```
chown root:lp /dev/usb/lp8
chmod 660 /dev/usb/lp8
```

- 如何使用 lm_sensors 侦测主机内的温度，详细说明整个步骤？
    - 先确定您的主机板具有温度与电压等侦测芯片，可使用 lspci 检查芯片组；
    - 开机进入 BIOS 后，查询一下是否具有硬件侦测温度、电压的项目，将输出的项目顺序记一下；
    - 确定 Linux 已经安装了 lm_sensors，再使用 sensors-detect 检查所需要的设定项目；
    - 依据上个步骤，设定 /etc/modprobe.conf 及 /etc/rc.d/rc.local 两个档案；
    - 使用 chkconfig 让 lm_sensors 开机启动，并且重新开机 (reboot)；
    - 开始使用 sensors 进行侦测，也可以尝试修改 /etc/sensors.conf 的内容，以符合实际状况。
- 我原本的 Linux 系统使用 80GB 的硬盘，分成 /dev/hda1(/), /dev/hda2(/home), /dev/hda3(swap)， 现在我想要将所有的数据通通搬移到另一颗 250GB 的硬盘上面去，所有的数据都不要改变， 我想要利用 dd, fdisk, mke2fs, resize2fs 等指令的辅助，可以如何工作？

  这个问题很有趣喔！建议你可以先参考这一篇：
  http://ms.ntcb.edu.tw/~steven/article/dd-sys-backup.htm ，主要的工作可以这样做：

    - 先以 fdisk 将 250GB 的硬盘分割成为 3 个 partitions ， 个别对应到 /dev/hda1, /dev/hda2, /dev/hda3 ，必须要注意， 后来的新硬盘的 partition 必须要大于原本的！
    - 利用 mke2fs 将后来新硬盘的 1, 2 partitions 格式化！
    - 利用『 dd if=/dev/hda1 of=/dev/hdb1 』将数据开始复制！
    - 使用 fsck 检查 /dev/hdb1, /dev/hdb2 两个 partition；
    - 利用本章學到的 resize2fs 去校正剩下的磁碟空間：『resize2fs /dev/hdb1』
    - 用盡各種方法將 grub 植入 /dev/hdb 當中！成功！搞定！ ^_^

Linux 核心编译与管理

最近更新日期：2005/12/05

我们在前面的几个章节介绍了 Linux 其实指的就是核心而已，而整个 Linux 的世界当中， 最重要的也就是核心了！他控制了您的服务器的所有硬件，也控制了所有的您想要的功能， 例如软件磁盘阵列(RAID)、各种适配卡的驱动模块、防火墙的新增功能等等。透过管理您的 Linux 核心，将可让您的服务器跑得更加的顺畅，也更稳定您的服务器所提供的相关服务呢！ 此外，为了让硬件驱动程序与相关的核心功能修订容易，所以 Linux 核心是支持『模块化』的，也就是说，您核心所想要的功能可以『后来才加挂上去』喔！那么如何加挂上去呢？这就需要了解一下所谓的『模块』了！当然，连同模块的相关指令就得也要会用啰！^_^

前言：

我们在第一章 Linux 是什么里面就谈过 Linux 其实指的是核心！ 这个『核心 (kernel)』是整个操作系统的最底层，他负责了整个硬件的驱动， 以及各个核心工具的提供，包括防火墙机制、是否支持 LVM 或

Quota 等档案系统等等， 这些都是核心所负责与提供的！也就因为如此，所以开机过程当中，除了 MBR 之外， 第一个读入系统内存当中的，就是核心档案。

如果你还记得开机流程里面提到的相关信息的话， 那么就知道，基本上，核心就是用来控制您的硬件系统的，您想要硬件帮您达成的任何工作， 都得要透过『核心』来帮您达成啊！换句话说，如果您的『核心』并没有支持您的某项硬设备， 也就是核心无法『认识』您的某项硬件，呵呵！那么该硬件也就无法在这样的核心底下工作了！

什么是核心(Kernel)

这已经是整个 Linux 基础的最后一篇了，所以，底下这些数据您应该都要『很有概念』才行～ 不能只是『好像有印象』～～好了，那就复习一下几个名词吧！

- Kernel：

还记得我们在前面的 BASH shell 部分提到过：计算机真正在工作的东西其实是『硬件』， 例如数值运算要使用到 CPU、数据储存要使用到硬盘、图形显示会用到显示卡、 音乐发声要有音效芯片、连接 Internet 可能需要网络卡等等。那么如何控制这些硬件呢？ 那就是核心的工作了！也就是说，你所希望计算机帮你达成的各项工作， 都需要透过『核心』的帮助才行！ 当然啰，如果你想要达成的工作是核心所没有提供的， 那么你自然就没有办法透过核心来控制计算机使他工作啰！

举例来说，如果你想要有某个网络功能 (例如核心防火墙机制) ， 但是你的核心偏偏忘记加进去这项功能，那么不论你如何『卖力』的设定该网络套件，很抱歉！ 不来电？换句话说，你想要让计算机进行的工作，都必须要『核心有支持』才可以 ！！这个标准不论在 Windows 或 Linux 这几个操作系统上都相同！如果有一个人开发出来一个『全新的硬件』，目前的核心不论 Windows 或 Linux 都不支持，呵呵！那么不论你用什么系统，哈哈！ 这个硬件都是英雄无用武之地啦！那么是否了解了『核心』的重要了呢？没错！所以我们才需要来了解一下如何编译我们的核心啦！

那么核心到底是什么啊？讲了这么多？？其实核心就是系统上面的一个档案而已，这个档案包含了驱动主机各项硬件的侦测程序与驱动模块，在 开机流程分析 章节当中，我们也提到这个档案被读入主存储器的时机是：

1. BIOS
2. MBR 载入 Loader (Linux 中的 Lilo 或 Grub 或 SPFdisk 等等)
3. 藉由 Loader 的辅助，加载核心档案到主存储器当中，此时核心档案解压缩后， 会开始侦测硬件的各项配备，并加载适当的驱动模块来让硬件生效；
4. 在硬件准备妥当后，加载第一支程序 init ，并藉由 /etc/inittab 的设定来确认预设 run level；
5. 经由 /etc/inittab 及 run level 来决定执行的各项启动的 scripts ；
6. 开始执行 login 或 X Window 等待登入等。

看到了吗？第三个步骤时，我们的系统就可以经由 loader 来将核心加载主存储器当中， 并且开始侦测分析硬件，加载适当的驱动程序，整个主机就可以待命，让使用者来执行相关的程序了。 这个核心档案通常被放置成 /boot/vmlinuz ，不过也不见得， 因为一部主机上面可以拥有多个核心档案，只是开机的时候

仅能选择一个来加载而已。 甚至我们也可以在一个 distribution 上面放置多个核心，然后以这些核心来做成多重开机呢！

- 核心模块：

还记得我们之前谈到开机流程分析时，提到的核心模块吧？ 既然核心档案都已经包含了硬件侦测与驱动模块，那么什么是核心模块啊？ 要注意的是，现在的硬件更新速度太快了，如果我的核心比较旧， 但我换了新的硬件，那么，这个核心肯定无法支持！怎么办？重新拿一个新的核心来处理吗？ 开玩笑～核心的编译过程可是很麻烦的～

所以啰，为了这个缘故，我们的 Linux 很早之前就已经开始使用所谓的模块化设定了！ 亦即是将一些不常用的类似驱动程序的咚咚独立出核心，编译成为模块， 然后，核心可以在运作的过程当中加载这个模块到核心的支持当中。 如此一来，我在不需要更动核心的前提之下，只要编译出适当的驱动模块，并且加载他， 呵呵！我的 Linux 就可以使用这个硬件啦！简单又方便！！

那我的模块放在哪里啊？？可恶！当然一定要知道的啦！就是 /lib/modules/`uname -r`/ 当中啦！

- 核心编译：

刚刚上面谈到的核心其实是一个档案，那么这个档案怎么来的？呵呵～ 当然是透过原始码 (source code) 编译而成的啊！因为核心是直接被读入到主存储器当中的， 所以当然要将他编译成为系统可以认识的数据才行！也就是说， 我们必须要取得核心的原始码，然后利用 Source code 与 Tarball 章节当中提到的编译概念来达成核心的编译才行啊！(这也是本章的重点啊！ ^_^)

- 关于驱动程序与核心的问题：

既然核心与硬件是息息相关的，那么是否意味着每次有厂商推出新版本的硬件时， 我们都需要『重新编译核心』啊？好加在！并不需要的喔！因为我们 Linux 的核心相当的具有弹性，他是支持模块化的，也就是说，只要新硬件可以推出搭配核心的驱动模块(也就是大家口头上常常讲的『驱动程序』)，那么我们只要将该模块挂加载核心，核心就可以支持该硬件啦！

但是，很多朋友还是常常感到困惑，就是 Linux 上面针对最新硬件的驱动程序总是慢了几个脚步， 所以觉得好像 Linux 的支持度不足！其实不可以这么说的，为什么呢？因为在 Windows 上面，对于最新硬件的驱动程序需求，基本上， 也都是厂商提供的驱动程序才能让该硬件工作的，因此，在这个『驱动程序开发』的工作上面来说， 应该是属于硬件发展厂商的问题，因为他要我们买他的硬件， 自然就要提供消费者能够使用的驱动程序啦！所以，如果大家想要让某个硬件能够在 Linux 上面跑的话，那么似乎可以发起一人一信的方式，强烈要求硬件开发商发展 Linux 上面的驱动程序！这样一来，也可以促进 Linux 的发展呢！

我干嘛要更新核心

这个『核心』是除了 BIOS 之外，一个操作系统中最早被启动的东西， 他包含了所有可以让硬件与软件工作的信息，所以，如果没有搞定核心的话， 那么你的系统肯定会有点小问题！好了，那么是不是将『 所有目前核心有支持的东西都给他编译进去我的核心中， 那就可以支持目前所有的硬件与可执行的工作啦！』！

这话说的是没错啦，但是你是否曾经看过一个为了怕自己今天出门会口渴、会饿、会冷、会热、会被车撞、

会摔跤、会被性骚扰，而在自己的大包包里面放了大瓶矿泉水、便当、厚外套、短裤、防撞钢梁、止滑垫、电击棒....等一大堆东西，结果却累死在半路上的案例吗？当然有！但是很少啦！我相信不太有人会这样做！（ 会这么做的人通常都已经在医院了～ ）取而代之的是会看一下天气，冷了就只带外套， 热了就只带短衣、如果穿的漂亮一点又预计晚点回家就多带个电击棒、 出远门到没有便利商店的地方才多带矿泉水....

说这个干什么！对啦！就是要您了解到，核心的编译重点在于『你要你的 Linux 作什么？』，是啦！如果没有必要的工作，就干脆不要加在你的核心当中了！这样才能让你的 Linux 跑得更稳、更顺畅！这也是为什么我们要编译核心的最主要原因了！

Tips:
说到这里突然想到以前国军研究的『经国号战斗机』事件，在当时，经国号里头的配备都是『最棒的！』包括测量仪器、瞄准配备、武器系统等等，但是呢， 却配上一部普普通通的客机用引擎！挖哩ㄌㄟ?最早期试飞的时候， 经国号竟然只能『在跑道上滑行！』真是悲哀！同样的道理， Linux 的核心也是这样的，如果你的硬件与核心之间没有办法达到良好的配合，那么 Linux 确实可能会跑得不很顺畅！！



Linux 的核心有几个主要的特色，除了『Kernel 可以随时、随各人喜好而更动』之外，Kernel 的『版本更动次数太频繁』也是一个特点！所以啰，除非你有特殊需求， 否则一次编译成功就可以啦！不需要随时保持最新的核心版本，而且也没有必要（ 编译一次核心要粉久的ㄋㄟ！ ）。话说到这里又突然想到今天看到的一篇文章，大意是说老板想要雇用的人会希望是 Linux 的老手，因为他们比较容易了解问题的所在，除此之外，如果有任何问题发生，由于其使用 Linux 是可以随时修补漏洞的！但是如果是 Windows 的话，就得要将机器关闭，直到 MS 推出修补套件后才能再启用～

那么是否『我就一定需要在安装好了 Linux 之后就赶紧给他编译核心呢？』， 老实说，『并不需要的』！这是因为几乎在每一个 distribution 当中，他们已经预设好了相当大量的模块了， 所以几乎使用者常常或者可能会使用到的数据都已经被编译成为模块，也因此，呵呵！ 我们使用者确实不太需要重新来编译核心！尤其是『一般的使用者， 由于系统已经将核心编译的相当的适合一般使用者使用了，因此一般入门的使用者，基本上， 不太需要编译核心』。

OK！那么鸟哥闲闲没事干跑来写个什么东西？既然都不需要编译核心还写编译核心的分享文章， 鸟哥卖弄才学呀！？很抱歉，鸟哥虽然是个『不学有术』的混混，却也不会平白无故的写东西要您来指教～ 当然是有需要才会来编译核心啦！编译核心的时机可以归纳为几大类：

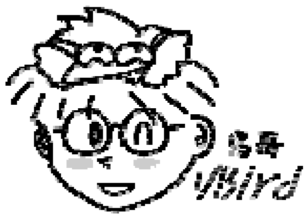
- 新功能的需求：
  我需要新的功能，而这个功能只有在新的核心里面才有，那么为了获得这个功能，只好来重新编译我的核心了（ 例如 iptables 这个防火墙机制只有在 2.4.xx 版本里面才有，而新出产的 AGP 显示卡，很多也需要新的核心推出之后，才能正常而且有效率的工作！）再举个例子， 之前的 Red Hat 7.2 的版本中，由于预设是将 CD-ROM 编译成核心的『模块』， 也就是说，核心本身还没有支持 CD-ROM 的功能，必须要挂上模块之后才能使用与读取这个 CD-ROM ！是否觉得很麻烦呢？呵呵！那么这个时候，如果你想要直接让 kernel 支持 CD ROM 的话，就得要重新编译核心啰；
- 原本核心太过臃肿：
  如果您是那种对于系统『稳定性』很要求的人， 那么对于核心多编译了很多莫名其妙的功能而不太喜欢的时候， 那么就可以重新编译核心来取消掉该功能啰；

- 与硬件的搭配稳定性：
  由于原本 Linux 的核心大多是针对 Intel 的 CPU 来作开发的，所以如果你的 CPU 是 AMD 的系统时，有可能 (注意！只是有可能，不见得一定会如此) 会让系统跑得『不太稳！』就鸟哥的经验来看，使用旧的 Pentum 系列的旧机器安装 Linux 的结果，还没有胡乱当机的经验！但是安装在 K6-2, K6-3 上面的 Linux ，通常需要重新编译一下核心会比较稳定一些！
- 其它：
  就是你需要特殊的环境需求时，就得自行设计你的核心啰！(像是一些商业的软件包系统，由于需要较为小而美的操作系统， 那么他们的核心就需要更简洁有力了！)

另外，需要注意重新编译核心虽然可以针对你的硬件作最佳化的步骤 (例如刚刚提到的 CPU 的问题！)，不过由于这些最佳化的步骤对于整体效能的影响是很小很小的， 因此如果是为了增加效能来编译核心的话，基本上，效益不大！然而，如果是针对『系统稳定性』来考虑的话， 那么就有充分的理由来支持您重新编译核心啰！

『如果系统已经运行很久了，而且也没有什么大问题， 加上我又不增加冷门的硬设备，那么建议就不需要重新编译核心了！』， 因为重新编译核心的最主要目的是『想让系统变的更稳！』既然您的 Linux 主机已经达到这个目的了，何必再编译核心？不过，就如同前面提到的， 由于预设的核心不见得适合您的需要，加上预设的核心可能并无法与您的硬件配备相配合， 此时才开始考虑重新编译核心吧！

Tips:
早期鸟哥是强调最好重新编译核心的一群啦！不过，最近这个想法改变了～ 既然原本的 distribution 都已经帮我们考虑好如何使用核心了，那么， 我们也不需要再重新的编译核心啦！尤其是 distribution 都会主动的释出新版的核心 RPM 版本， 所以，实在不需要自己重新编译的！当然啦，如同前面提到的， 如果您有特殊需求的话，那就另当别论噜！ ^_^



然而由于『核心的主要工作是在控制硬件！』所以编译核心之前， 请先了解一下您的硬件配备，与您这部主机的未来功能！由于核心是『 越简单越好！』所以只要将这部主机的未来功能给他编进去就好了！ 其它的就不用去理他啦！

核心的版本与何处下载最新核心

既然这一章的内容要讨论的是如何编译核心，那么我们就来聊一聊核心的版本吧！

- 核心的版本：

由于不同的核心版本之间，使用的函式库并不相同，所以，我们必须要知道自己的 Linux 核心版本之后，才能够取用最新支持的核心版本来编译新的核心啊！那么要在哪里找到这个核心的信息呢？ 应该还记得那个 uname 查看的信息吧！没错！就是他了，可以使用其功能来查询目前在工作的核心版本：

```
[root@linux ~]# uname -r
2.6.13-1.1532_FC4
# 因为鸟哥的 FC4 已经升级核心多次，所以这个版本应该与你的不同！
```

看到了吧！那个东西就是核心版本的信息啦！好了！我们依照 RPM 版本的先例，也来谈一谈 kernel 的版本吧！基本上， kernel 的版本可以区分为：

[主版本].[次版本].[释出版本(release)]-[修改版本]

整个版本的定义当中，最需要注意的是前两个，亦即主版本与次版本。 相同的[主][次]版本，代表他使用的函式库是差不多的，所以，可以直接升级到较高的[释出版本]上。 值得注意的是，由于核心功能的增加速度实在太快了，一般商业用户与一般使用者， 根本不需要很多的测试中的功能，因此，[主][次]版本中，依据[次版本]的奇偶数， 又分为底下两种版本：

- 如果[次版本]是奇数的话，例如 2.3, 2.5 等等，那表示他是一个『 测试性质功能的核心版本 』，这种核心通常是在推出稳定版本的核心之前，用来给 developer（ 核心维护更新测试者！） 测试用的！虽然功能较为强大，但是由于是属于测试性质，所以可能会有些许的 bugs 也说不定；
- 如果[次版本]是偶数的话，例如 2.4, 2.6 等等，那表示他是一个经过测试之后才释出的 『稳定核心版本，这种核心较为稳定不容易出错， 比较适合一般个人或者是商业使用！

所以啦！我们要升级的时候，大多就是使用那种偶数的核心版本啦！不过这里还是要再提一遍！就是『 2.4 与 2.6 是两个具有相当大差异的核心版本， 两者之间使用到的函式库基本上已经不相同了，所以在升级之前，如果您的核心原本是 2.4.xx 版，那么就升级到 2.4.xx 版本的最新版，不要由 2.4.xx 直接升级到 2.6.xx 版，否则到时可能会欲哭无泪～～』， 这个问题在讨论区一再地被提起！这里再次说明！

Tips:
为什么不能从 2.4 升级到 2.6 呢？其实还是可以啦！只是过程很复杂！ 我们知道软件 (packages) 是架构在系统核心上面来进行编译、安装与执行的， 也就是说，这些 packages 与核心之间，是有相关性的！这些 packages 会用到很多核心提供的功能。 但是不同的[主][次]版本之间，他们提供的功能架构差异太大，因此，若你由 2.4 升级到 2.6 的话， 那么绝大部分的软件『都需要重新再编译！』这样了解为何不要在不同的版本间升级了吧？



此外，2.4.xx 与 2.6.xx 的比较中，并不是 2.6.xx 就一定比 2.4.xx 还要新，因为这两种版本同时在进行维护与升级的工作！如果有兴趣的话，可以前往 Linux 核心网站 http://www.kernel.org 一看究竟，您就可以了解目前的核心变动情况了！

基本上，目前最新的 distributions ，包括 FC, SuSE, Mandriva 等等，都使用 2.6 的核心， 所以，您可以直接由 http://www.kernel.org 下载最新的 2.6.xx 版本的核心来尝试编译啊！目前 (2005/11/20) 鸟哥可以查到的最新版本是 2.6.14-2 ， 底下我们将主要以这个版本来测试。另外，由于较新的核心版本可能会多出一些选项， 因此若有不同的项目也没有关系！稍微查看一下说明内容就可以了解啦！

例题：什么是『释出版本』？
答：

由于核心的新功能增加太快，为了要统合这些功能，因此，每隔一段时间的稳定性测试后， 这些新功能才会被放到原本的核心内，最后被推出。而为了与前一个核心原始码作区别， 所以就被加上一个数字较高的『释出版本』数字了。

例题：那什么是『修改版本』？
答：

由于原本的核心原始码可能有点 bugs 在里面，经过程序开发人员的程序代码修改后 (debug)， 再重新推出的一个类似加强版的意思。基本功能是不变的，只是有问题的地方被克服而已。

- 核心下载地点：

Linux 的核心目前是由其发明者 Linus Torvalds 所属团队在负责维护的， 而其网站在底下的站址上，在该网站上可以找到最新的 kernel 信息！不过，美中不足的是目前的核心越来越大了(linux-2.6.14.2.tar.bz2 这一版，这一个档案大约 37MB 了！)，所以如果你的 ISP 连外很慢的话， 那么使用台湾的映射站台来下载不失为一个好方法：

- http://www.kernel.org/
- 交大资料：ftp://linux.cis.nctu.edu.tw/kernel/
- 义守大学：http://ftp.isu.edu.tw/pub/Linux/kernel/

核心原始码的取得与升级

既然核心是个档案，要制作这个档案给系统使用则需要编译，既然要有编译，当然就得要有原始码啊！ 那么原始码怎么来？除了刚刚前一个小节提到的，需要注意核心的版本之外，还有哪些要注意的事项？？

取得原本的 distributions 提供的 kernel source

事实上，各大主要 distributions 在推出他们的产品时，其实已经都附上了核心原始码了！ 以我们的 FC4 为例，你如果有安装工具程序的话，那么应该就可以利用 rpm 找到套件名称为 kernel-devel 的套件，那就是我们的核心原始码了(这个套件名称在各个不同的版本上头都不一样！ 所以，您应该要使用 rpm -qa | grep kernel 来寻找喔!)。如果还是找不到，那表示你没有安装啊！ 此时，拿出原版光盘，一片一片去 mount 且搜寻一下，肯定可以找到的啦！ 然后安装他就好了！

既然要重新编译，那么干嘛还要使用原本 distributions 释出的原始码啊？真没创意～ 话不是这么说，因为原本的 distribution 释出的原始码当中，含有他们设定好的预设设定值， 所以，我们可以轻易的就了解到当初他们是如何选择与核心及模块有关的各项设定项目的参数值， 那么就可以利用这些可以配合我们 Linux 系统的预设参数来加以修改，如此一来， 我们就可以『修改核心，调整到自己喜欢的样子』啰！ 而且编译的难度也会比较低一点！

取得最新的核心

虽然使用原本的 source code 来重新编译核心比较方便，但是，如此一来， 新硬件所需要的新驱动程序，也就无法藉由原本的核心原始码来编译啊！ 所以啰，如果是站在要更新驱动程序的立场来看，当然使用最新的核心会比较好啊！

取得最新的核心版本，上一个小节已经讲过了，请自行前往 http://www.kernel.org 去下载吧！

保留原本设定：利用 patch 升级核心原始码

如果你曾经自行以最新的核心版本来编译过核心，那么你的系统当中应该已经存在前几个版本的核心原始码， 以及上次你自行编译的参数设定值才对。如果您只是想要更新到最新版本的核心， 原本的参数设定值并不要进行大幅度的修改，那么该如何是好？

呵呵！每一次核心释出时，除了释出完整的核心压缩档之外，也会释出『该版本与前一版本的差异性 patch 档案』， 关于 patch 的制作我们已经在 原始码与 tarball 章节当中提及， 您可以自行前往参考。这里仅是要提供给您的是，每个核心的 patch 仅有针对前一版的核心来分析而已， 所以，万一你想要由 2.6.10 升级到 2.6.14 的话，那么你就得要下载 2.6.11, 2.6.12, 2.6.13 及 2.6.14 的 patch file ，然后『依次』一个一个的去进行 patch ， 才能够升级到 2.6.14 喔！这个重要！不要忘记了。

在进行完 patch 之后，你可以直接检查一下原本的设定值，如果没有问题， 就可以直接编译，而不需要再重新的选择核心的参数值，这也是一个省时间的方法啊！ 至于 patch file 的下载，同样是在 kernel 的相同目录下，寻找文件名是 patch 开头的就是了。

核心目录下的次目录信息

假设你已经有安装了核心的原始码， 以 FC4 为例，他的预设核心原始码放置在 /usr/src/kernels/2.6.11-1.1369_FC4-i686/ 这个目录下，在该目录下，基本上有这几个目录：

```
arch       ：与硬件平台有关的项目，例如 CPU 的等级等等；
crypto     ：核心所支持的加密的技术，例如 md5 或者是 des 等等；
drivers    ：一些硬件的驱动程序，例如显示卡、网络卡、PCI 相关硬件等等；
fs         ：核心所支持的 filesystems ，例如 vfat, reiserfs, nfs 等等；
lib        ：一些函式库；
net        ：与网络有关的各项协议数据，还有防火墙模块 (net/ipv4/netfilter/*) 等等；
sound      ：与音效有关的各项模块；
```

每个目录底下也都含有很多不同的次目录，例如 drivers 目录下就含有 net, sound, usb, pci, vedio... 等等多到数不清的次目录，这些目录底下还是含有相关的硬件驱动模块等等～呵呵～想要完整的了解是很难的啦！ 例如，核心使如何让工作排到 CPU 去执行的？核心是如何存取物理内存与 Swap ？ 核心是如何读取各不同的 filesystems 等等，如果有兴趣的话， 那么最新核心档案解压缩之后，都会有个 Documentation 的目录， 可以进去查阅各个相关的说明啊！ ^_^

设定核心的编译设定 (Makefile)

就如同我们在原始码与 tarball 的章节当中提到的， 由于各个主机硬件都不相同，所以当然需要针对我们的主机环境来选择可以编译的项目啦！那就是 Makefile 的编辑。但是核心的数据实在多到不行～所以，核心有提供不少的工具来让我们简单的进行参数的设定喔！

如何编辑核心的 Makefile

在这一章当中，鸟哥假设你是以 http://www.kernel.org 这个核心官方网站下载最新的核心版本来编译的， 鸟哥下载的是 2.6.14-2 版，下载的完整网址在：

    http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.14.2.tar.bz2

如果你是以 FC4 系统预设的核心原始码来重新编译， 那么请自行安装 kernel-devel 这个套件，以及前往 /usr/src/kernels/ 目录下， 找到核心原始码；如果您之前已经以下载的核心档案编译过核心， 那么

请依序下载各 patch 档案，然后请自行参考 patch 的用法以及找到相关的路径吧！反正，这一章当中，我假设您与我一样， 使用的是最新版的核心就是了。

假设你下载之后将整个档案放置到 /root 内，那么首先请解压缩吧！

```
[root@linux ~]# cd /usr/src
[root@linux src]# tar -jxvf /root/linux-2.6.14.2.tar.bz2
# 这个时候就会产生一个 /usr/src/linux-2.6.14.2 的目录，该目录就是 source code。
# 不过，这个目录下有个 README 的档案务必参考，此外，
# 还有个 Documentation 的目录，也可以仔细的看一看喔！

[root@linux src]# cd linux-2.6.14.2
[root@linux linux-2.6.14.2]# make mrproper
# 这个过程在删除一些以前留下来的 .o 档案。
```

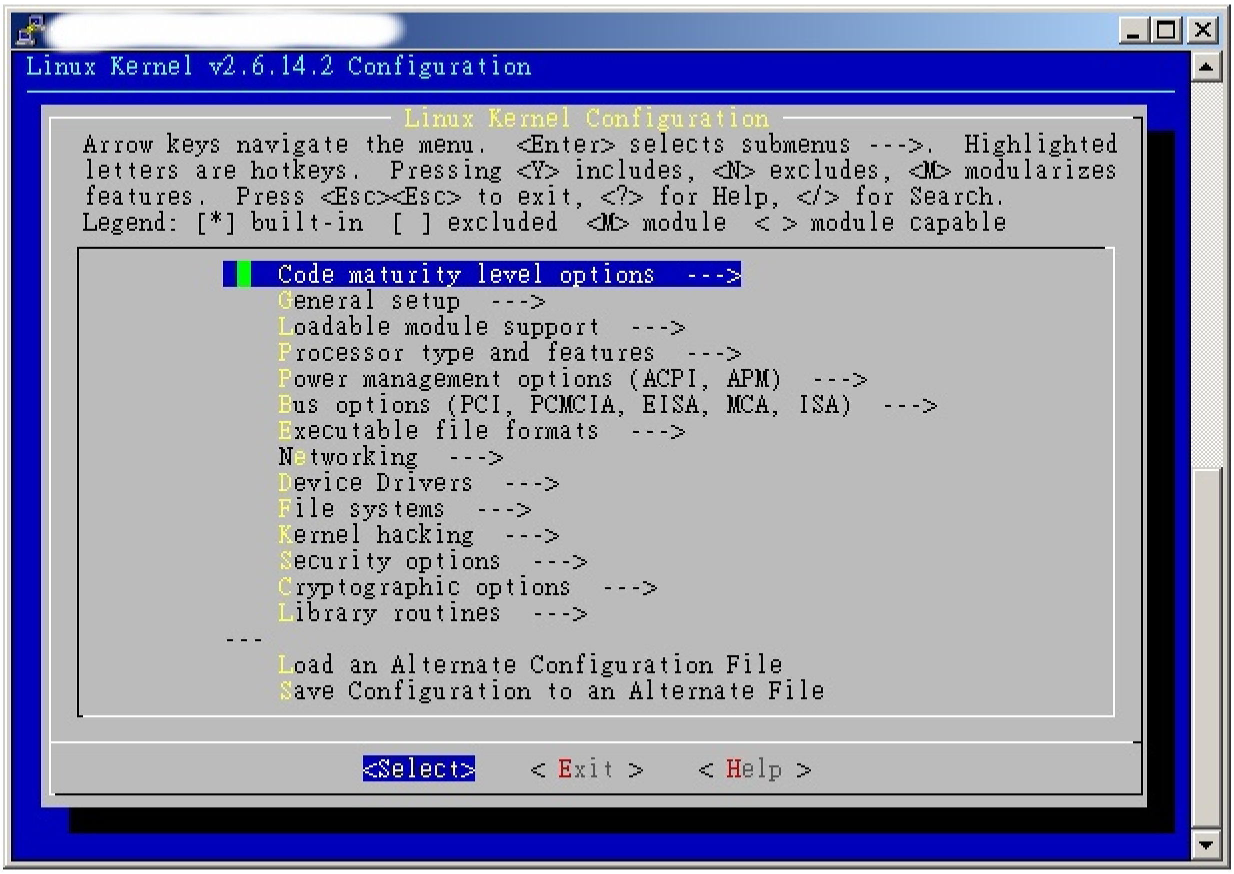
特别留意一下，核心的原始码最好不要直接放置到 /usr/src/linux 这个目录， 这是因为该目录是很多的软件读取核心功能的目录，如果你将这个新核心放置到 /usr/src/linux 下时， 可能会让某些软件读到错误的核心档案。因此，您才可能看到目前的核心原始码都放到 /usr/src/kernels/ 目录下 (FC4 的预设放置目录)。 所以，我们新的核心主要建议您还是放置到 /usr/src/ 目录下， 但是目录名称保持 linux-2.6.14.2 即可，不必更名为 linux 啰！这点重要！

另外，在进行核心的参数设定之前，务必要执行『 make mrproper 』这个项目， 还记得原始码的编译过程吧？编译过程会有 *.o 的目标档案对吧！ 那这些 *.o 必须要先删除啊！否则可能会产生旧的数据啊！这个要注意。

另外，其实我们也知道，整个原始码的编译过程当中，那个 Makefile 占有举足轻重的地位的！ 这是因为我们的 make 取用的参数数据都是记录在 Makefile 档案当中啊！ 所以啰，你必须要确定你的系统已经安装了 make, gcc 等等的编译套件， 否则是无法进行编译的。此外，核心的 Makefile 没有办法像一些软件一样， 简单的使用 ./configure 就能够自动的侦测主机。这是因为每个人对于核心的要求都不一样嘛！好了，那么如何建立 Makefile 啊？难道要手动去编辑？？当然不是啦！ 我们可以透过核心提供的功能，就是那个 make menuconfig 来达成喔！

- make menuconfig:
  利用类似选单模式的方式来进行核心参数的挑选，好处是，他是纯文字模式的！ 不需要启动 X Window ，还可以远程登入进行核心参数的挑选！真方便！
- make xconfig:
  利用 X Window 的功能来进行挑选，是图形接口的，很华丽～不过，当然就比较耗系统资源。 如果你的服务器没有安装 X Window ，那就别提了！
- make gconfig:
  利用 GDK 函式库的图形接口来选择，也是需要 X Window 的支持才行！

还有一些早期的编译流程，不过不好用，所以鸟哥就不介绍了。我这里推荐您使用 make menuconfig 来进行核心参数的挑选。这是因为很多的服务器本来就可能没有 X Window ， 加上 make menuconfig 也可以作类似图形化接口的选单模式，可以随时作参数的选择， 方便又好用！ ^_^。只要在 /usr/src/linux-2.6.14.2 目录下，输入『make menuconfig』 就可以出现如下的画面喔！



图、核心编译工作前的参数挑选

看到上面的图是之后，你会发现主要分为两大画面，一个是大框框内的反白光柱，另一个则是底下的小框框， 里面有 select, exit 与 help 三个选项的内容。这几个组件的用法如下：

- 最底下的 <Select> <Exit> <Help> ：可以使用『左右键』来移动光标；
- 上下键可以移动上面大框框部分的 Code maturity level options 那一行！ 若该行有箭头『 ---> 』则表示该行内部还有其它细项需要来设定的意思；
- 当以『上下键』选择好想要设定的项目之后，并以『左右键』选择 <Select> 之后， 按下『 Enter 』就可以进入该项目去作更进一步的细部设定啰；
- 在细部项目的设定当中，如果前面有 [ ] 或 < > 符号时，该项目才可以选择， 而选择可以使用『空格键』来选择；
- 若为 [*] <*> 则表示编译进核心；若为 <M> 则表示编译成模块！ 尽量在不知道该项目为何时，且有模块可以选，那么就可以直接选择为模块啰！
- 当在细项目选择 <Exit> 后，并按下 Enter ，那么就可以离开该细部项目啰！

基本上建议只要『上下左右 空白 及 Enter 』这六个按键就好了！不要使用 Esc ，否则一不小心就有可能按错的！另外，关于整个核心的内容选择上面，建议您可以这样思考：

- 『肯定』核心一定要的功能，直接编译进核心内；
- 『可能在未来会用到』的功能，那么尽量编译成为模块；
- 『不知道那个东西要干嘛的，看 help 也看不懂』的话，那么就保留默认值，或者将他编译成为模块；

总之，尽量保持核心小而美，剩下的，就编译成为模块，尤其是『需要考虑到未来扩充性』， 像鸟哥之前认为螃蟹卡就够我用的了，结果，后来竟然网站流量大增，鸟哥只好改换 3Com 的网络卡。 不过，我的核心却没有相关的模块可以使用～因为.....鸟哥自己编译的核心忘记加入这个模块了。 最后，只好重新编译一次核心的模块，呵呵！真是惨痛的教训啊！

核心的内容与模块设定

由上面的图示当中，我们知道核心的可以选择的项目有很多啊！ 光是第一面，就有 17 个项目，每个项目内还有不同的细项！哇！真是很麻烦啊～ 而每个项目其实都可能有 <Help> 的说明，所以，如果看到不懂的项目， 务必要使用 Help 查阅查阅！好了，底下我们就一个一个项目来看看如何选择吧！

- Code maturity level options(核心的 code 开发维护)

这个项目主要在设计您的核心是否要支持一些尚未测试的很完整功能。 一般来说，我们是一般用户，不是 kernel 的开发维护者，所以，当然不需要额外的功能啦！ 所以，鸟哥这里的选择是比较保守的 (不使用额外功能)，也因为如此， 所以底下的很多项目当中，可能不会出现一些较为特殊的选项喔！这个要注意！鸟哥的选择如下：

```
[ ] Prompt for development and/or incomplete code/drivers
# 这个可选可不选～不过，鸟哥这里是不选择的啦！
```

- General setup

这个项目则是关于核心的一般设定，包括核心的附加版本信息等等，都可以在这里设定。

```
()  Local version - append to kernel release
[ ] Automatically append version information to the version string (NEW)
          # 上面这两个都与核心的附加版本有关。例如 FC4 的核心版本为
          # 2.6.14-1.1637_FC4 ，后面那个 1.1637_FC4 就是那个附加版本啦！
[*] Support for paging of anonymous memory (swap)
          # 这个与 swap 的使用有关！当然要选择啦
[*] System V IPC
          # IPC 是 Inter Process Communication 的简写，这个与一个 programs
          # 可以被多人同时启用有关，所以务必要选择才行！
[*] BSD Process Accounting
[ ]   BSD Process Accounting version 3 file format
[*] Sysctl support
          # 这个就是在产生 /proc/sys 的支持！务必选择！
[*] Auditing support
[*]   Enable system-call auditing support
          # 上面这两个是额外核心功能 (如 SELinux) 加载时所需要的设定！务必选择
--- Support for hot-pluggable devices
[*] Kernel Userspace Events
          # 让核心能够监听使用者的动作。举例来说， USB 装置的联机与否等等的实时装置。
[ ] Kernel .config support
```

```
()  Initramfs source file(s)
[ ] Configure standard kernel features (for small systems)  --->
```

这里的项目主要都是针对核心与程序之间的相关性来设计的，基本上， 保留默认值即可！除非您想要编辑属于自己的附加版本，那么在上表的第一项按下 Enter 后， 就可以输入一些信息了。不要随便取消上面的任何一个项目， 可能会造成某些程序无法被同时执行的困境喔！

- Loadable module support

还记得我们上头曾经提过模块这个玩意儿吧！如果你要核心能够支持模块实时加载某些核心功能的话， 那么这里面的设定就显的很重要了！因为他涉及是否支持模块加载啊！

```
[*] Enable loadable module support
[*]   Module unloading
[*]   Source checksum for all modules
[*]   Automatic kernel module loading
```

不用想太多，全部都选择吧！

- Processor type and features

这个就与您的 CPU 有关啦！我的主机装备的是 P-III 的 CPU ，所以就选择相关的即可。 你要依据你自己的主机来设计喔！不要胡乱选择啊！

```
Subarchitecture Type (PC-compatible)  --->
          (X) PC-compatible    <==这里是次目录
          ( ) AMD Elan
          ( ) Voyager (NCR)
          ( ) NUMAQ (IBM/Sequent)
          ( ) SGI 320/540 (Visual Workstation)
          # 这里在选择主机的硬件类型。我们使用 PC 兼容的主机啊！选这个就对了。

Processor family (Pentium-III/Celeron(Coppermine)/Pentium-III Xeon)  --->
          ( ) 386        <==这里是次目录
          ( ) 486
          ( ) 586/K5/5x86/6x86/6x86MX
          ( ) Pentium-Classic
          ( ) Pentium-MMX
          ( ) Pentium-Pro
          ( ) Pentium-II/Celeron(pre-Coppermine)
          (X) Pentium-III/Celeron(Coppermine)/Pentium-III Xeon
          ( ) Pentium M
          ( ) Pentium-4/Celeron(P4-based)/Pentium-4 M/Xeon
          ( ) K6/K6-II/K6-III
          ( ) Athlon/Duron/K7
          ( ) Opteron/Athlon64/Hammer/K8
```

```
          ( ) Crusoe
          ( ) Efficeon
          ( ) Winchip-C6
          ( ) Winchip-2
          ( ) Winchip-2A/Winchip-
          ( ) GeodeGX1
          ( ) CyrixIII/VIA-C3
          ( ) VIA C3-2 (Nehemiah)
          # 这里则是 CPU 的等级，我使用的是 P-III ，您得要选择自己的啊！

[*] Generic x86 support
          # 对 x86 的 CPU 架构支持较佳。
[*] HPET Timer Support
[ ] Symmetric multi-processing support
          # 如果您使用两颗 CPU 以上的系统，这里『务必』要选择！否则不用选
    Preemption Model (No Forced Preemption (Server))  --->
          (X) No Forced Preemption (Server)
          ( ) Voluntary Kernel Preemption (Desktop)
          ( ) Preemptible Kernel (Low-Latency Desktop)
          # 这里与 CPU 的效能有关。如果您想要作一个服务器的核心，选择第一个！
          # 否则，为了稳定，最好选择第二项。

[*] Local APIC support on uniprocessors
[*]   IO-APIC support on uniprocessors
          # 单颗 CPU 的环境中，这个项目可以选择起来，让 CPU 具有
          # Advanced Programmable Interrupt Controller 的功能啊！
[*] Machine Check Exception
          # 让 Pentium 系列的 CPU 可以在侦测到 kernel 有问题时，立刻响应到终端接口
< >   Check for non-fatal errors on AMD Athlon/Duron / Intel Pentium 4
< > Toshiba Laptop support
< > Dell laptop support
          # 上面这三个就得要看看你的系统是否支持啦！基本上，可以设定成 M 啦！
[ ] Enable X86 board specific fixups for reboot
<M> /dev/cpu/microcode - Intel IA32 CPU microcode support
<M> /dev/cpu/*/msr - Model-specific register support
<M> /dev/cpu/*/cpuid - CPU information support
          # 因为我的是 P-III CPU 啊，所以这里当然选择成模块即可！
    Firmware Drivers  --->
       < > BIOS update support for DELL systems via sysfs (NEW)
       < > Dell Systems Management Base Driver (NEW)
          # 如果你的系统是 Dell 的，那么上面记得编成模块！鸟哥不需要～
    High Memory Support (4GB)  --->
          ( ) off
```

```
        (X) 4GB
        ( ) 64GB
        # 这个重要！一般来说，我们对于主机的要求是 RAM 越大越好(一般情况下)；
        # 但是，原本的核心支持仅到 1GB 的内存，所以，这里要加大！
        # 一般的个人计算机主机，或者是 X86 主机，通常只要 4GB 就够了，
        # 除非是特殊的工业用主机才可以额外插到 4GB 以上的内存！
        # 如果这里选择成 off 的话，那么您的内存最大只能被捉到 1GB 。

[*] Allocate 3rd-level pagetables from highmem
        # 这个与 High Memory Support 有关，如果你的内存支持到 4GB，这里可以加入
[ ] Math emulation
        # 这个与 CPU 是否具有浮点运算单元有关。目前我们的 CPU (586 以上)
        # 都已经内建了浮点运算单元了，所以这里可以不要选啦！
[*] MTRR (Memory Type Range Register) support
        # 这玩意儿可以让 CPU 具有读取内存特殊区块的能力，尤其在高效能的 AGP
        # 与相关的 PCI/AGP 总线进行数据传输时，可以增进不少效能。
        # 选择这个项目后，会产生 /proc/mtrr ，我们的 X 会读取这个咚咚喔。
[ ] Boot from EFI support (EXPERIMENTAL)
[*] Enable seccomp to safely compute untrusted bytecode
        # 这个项目通常要加，不过，如果是嵌入式系统的话，可以不加入！
    Timer frequency (250 HZ)
        # 这个项目则与核心针对某个事件立即响应的速度有关。一般来说，
        # 如果是一般桌上计算机，那么反应时间可以调整的快速一点，因为不会有其它事件。
        # 如果是主机，由于同一时间点可能有多人联机进来，启发的事件太多了，所以，
        # 这个反应时间反而要调慢一点，会比较稳定，而且效能也不差。通常保留默认值
        # 250 就很好了。
```

- Power management options (ACPI, APM)

这部分则是电源管理，主要的内容有底下这些：

```
[ ]  Power Management Debug Support
[ ] Software Suspend
        # 这个与将目前的环境暂存在 swap 当中有关。万一你想要将目前的资料暂存，
        # 因为系统可能必须要关机一阵子，那么这个项目可以选择。不过，
        # 由于可能会有一些问题，所以不建议您使用这个功能(主机也很少用到！)
    ACPI (Advanced Configuration and Power Interface) Support  --->
        # 这个电源管理模块虽然可以管理你的电源，不过，却会增加核心约 70K ，所以
        # 对嵌入式系统来说，可能需要考虑考虑。至于 desktop/server 当然就选择啊！
        [*] ACPI Support
        [*]   Sleep States
        [*]     /proc/acpi/sleep (deprecated)
        # 如果要启动 ACPI 的支持，那上面这几个几乎都是必要的！
        <M>   AC Adapter
        <M>   Battery
```

```
        <M>    Button
        <M>    Video
        # 这几个则只要编译成为模块即可，因为桌上型与服务器用不到。
        # 他主要大该都是针对笔记型计算机来设计的！ ^_^
        <*>    Fan
        <*>    Processor
        <*>      Thermal Zone
        # 每一部主机都有的 CPU/风扇 等，当然也可以编译进核心，也可以设定成模块。
        <M>    ASUS/Medion Laptop Extras
        <M>    IBM ThinkPad Laptop Extras
        <M>    Toshiba Laptop Extras
        (2001) Disable ACPI for systems before Jan 1st this year
        [ ]    Debug Statements
        [*]    Power Management Timer Support

APM (Advanced Power Management) BIOS Support  --->
        <*> APM (Advanced Power Management) BIOS support
        [ ]    Ignore USER SUSPEND
        [ ]    Enable PM at boot time
        [*]    Make CPU Idle calls when idle
        [ ]    Enable console blanking using APM
        [*]    RTC stores time in GMT
        [ ]    Allow interrupts during APM BIOS calls
        [ ]    Use real mode APM BIOS call to power off
        # 由于鸟哥比较少使用电源管理，所以，我这里大多使用默认值而已。

CPU Frequency scaling  --->
        # 什么？可以经过核心修改 CPU 的运作频率？哈哈！没错！是这样！
        # 不过，在说明档当中也提及，还需要启动底下的 dynamic cpufreq governor
        # 才可以顺利的启动这个项目。当然，如果你不愿意的话，这里可以取消。
        [*] CPU Frequency scaling
        [*]    Enable CPUfreq debugging
        <M>    CPU frequency translation statistics
        [*]      CPU frequency translation statistics details
        # 如果想要启动在休眠时， CPU 自动降频的功能，上面都给他设定好吧！
               Default CPUFreq governor (userspace)  --->
                    ( ) performance
                    (X) userspace
                    # 休眠时 CPU 频率的考虑，是以效能为主，还是您可以手动修改
                    # 既然要自动降频，当然不以效能为考虑～所以选 userspace 吧！

        <*>    'performance' governor
        <M>    'powersave' governor
```

```
        ---   'userspace' governor for userspace frequency scaling
        <M>   'ondemand' cpufreq policy governor
        <M>   'conservative' cpufreq governor
        # 上面这几个则是在载入哪些调节器(governor) ~
        ---   CPUFreq processor drivers
        <M>   ACPI Processor P-States driver
        < >   AMD Mobile K6-2/K6-3 PowerNow!
        <M>   AMD Mobile Athlon/Duron PowerNow!
        < >   Cyrix MediaGX/NatSemi Geode Suspend Modulation
        <*>   Intel Enhanced SpeedStep
        [*]     Use ACPI tables to decode valid frequency/voltage pairs
        [*]     Built-in tables for Banias CPUs
        <*>   Intel Speedstep on ICH-M chipsets (ioport interface)
        <M>   Intel Pentium 4 clock modulation
        <*>   Transmeta LongRun
        < >   VIA Cyrix III Longhaul
        # 上面这几个就与 CPU 的型号有关啦！我用的是 P-III，
        # 所以，不相关的数据我直接将他编成模块而已！
        ---   shared options
        [ ]   /proc/acpi/processor/../performance interface (deprecated)
        [ ]   Relaxed speedstep capability checks

        # 其实，这个项目主要是在主机 Idle 的时候，透过 CPU 本身的功能，
        # 然后让系统可以自动的降频的一个选项啦！ ^_^
```

老实说，由于鸟哥的 Linux 机器主要都是站在 Server 的角度，所以我的机器都是全年无休的。 在这样的条件下，我老是选择不要使用电源管理的说～ @_@。不过，如果是站在桌上型计算机的角度， 呵呵～启动电源管理这可是很棒的选项，因为.....电费越来越贵了～ 能省则省啊！ ^_^ 另外，绝大部分的选项都可以编译成为模块啊！只是会花去一些编译的时间就是了。

- Bus options (PCI, PCMCIA, EISA, MCA, ISA)

这个项目则与总线有关啦！分为最常见的 PCI，还有笔记型计算机常见的 PCMCIA 插卡啊！详细的资料有这些：

```
--- PCI support
      PCI access mode (Any)  --->
[ ]   PCI Express support
        # 这个重要！如果你的主机板有支持较新的 PCI-Express 显示卡的话，
        # 这里请务必要勾选～鸟哥的主机板太旧了，用的是 AGP 显示卡，所以这里不选！
[ ]   Message Signaled Interrupts (MSI and MSI-X)
[*]   Legacy /proc/pci interface
[ ]   PCI Debugging
[*] ISA support
[ ]   EISA support
```

```
         # 这个是比 PCI 还要更早的总线插槽，一般来说，
         # 最好还是保留 ISA 插槽比较妥当点～
[ ] MCA support
< > NatSemi SCx200 support
    PCCARD (PCMCIA/CardBus) support  --->
         < > PCCard (PCMCIA/CardBus) support
         [ ]   Enable PCCARD debugging
         < >   16-bit PCMCIA support
         [ ]     PCMCIA control ioctl (obsolete)
         ---   32-bit CardBus support
         ---   PC-card bridges
         < >   CardBus yenta-compatible bridge support
         < >   Cirrus PD6729 compatible bridge support
         < >   i82092 compatible bridge support
         < >   i82365 compatible bridge support
         < >   Databook TCIC host bridge support
         # 这个是 PC 卡，一般来说，桌上型计算机不会有这种卡的存在，
         # 所以，鸟哥通常是不选择～不过，如果你的主机是笔记型计算机，
         # 这里可就得要选择了！否则很多插卡就不能被使用啊！切记切记！

    PCI Hotplug Support  --->
         # 这个是进阶功能，可以不用理他！
```

PCI 插槽是重要的，因为几乎所有的适配卡都是插在 PCI 插槽上面的。 此外，这个设定项目里面有个比较有趣又重要的地方，那就是 PCI-E (PCI Express) 的设定项目了！如果你的主机板是最近买的，而且你的显示卡是 PCI-E 的话， 这个项目就务必要编入核心才行！否则显示卡会捉不到的！

- Executable file formats

这里必须要勾选才行喔！因为是给 Linux 核心运作执行文件之用的数据！ 除了第一项必须要编成核心功能之外，其它两项是可以编译成为模块的啦！

```
[*] Kernel support for ELF binaries
<M> Kernel support for a.out and ECOFF binaries
<*> Kernel support for MISC binaries
```

- Networking

这个项目是相当相当相当 * n 重要的选项，因为他还包含了防火墙相关的项目！ 就是未来在服务器篇会谈到的防火墙 iptables 这个数据啊！所以，千万注意了！

```
--- Networking support
      Networking options  --->
         # 就是这个光啊！里面的数据全部都是重要的防火墙项目！
         # 在这里面的项目当中，如果可以编成模块，尽量将他编成模块！
         <*> Packet socket
```

```
# 唯独这个项目务必要编进核心里面！因为他是防火墙啊！
[*]   Packet socket: mmapped IO
<*> Unix domain sockets
<*> IPsec user configuration interface
<M> PF_KEY sockets

# 底下是 TCP/IP 的设定，大多是 IPv4 ，只要保留默认值就很 OK 了！
[*] TCP/IP networking
[*]   IP: multicasting
[*]   IP: advanced router
        Choose IP: FIB lookup algorithm (choose FIB_HASH if unsure)
[*]   IP: policy routing
[*]     IP: use netfilter MARK value as routing key
[*]   IP: equal cost multipath
[ ]     IP: equal cost multipath with caching support (EXPERIMENTAL)
[*]   IP: verbose route monitoring
[ ]   IP: kernel level autoconfiguration
<M>   IP: tunneling
<M>   IP: GRE tunnels over IP
[*]     IP: broadcast GRE over IP
[*]   IP: multicast routing
[*]     IP: PIM-SM version 1 support
[*]     IP: PIM-SM version 2 support
[*]   IP: TCP syncookie support (disabled per default)
<M>   IP: AH transformation
<M>   IP: ESP transformation
<M>   IP: IPComp transformation
<M>   IP: tunnel transformation
<*>   INET: socket monitoring interface
[ ]   TCP: advanced congestion control

      IP: Virtual Server Configuration  --->
          # 这个项目则主要与 cluster 有关~里面保留默认值即可！

# 这底下则与 IPv6 ，新一代的 IP 协议有关！同样做成模块！
<M>   The IPv6 protocol
[*]     IPv6: Privacy Extensions (RFC 3041) support
<M>     IPv6: AH transformation
<M>     IPv6: ESP transformation
<M>     IPv6: IPComp transformation
---     IPv6: tunnel transformation
<M>     IPv6: IPv6-in-IPv6 tunnel
```

```
# 底下就重要啦！就是我们一直讲一直讲的防火墙啦！ ^_^
[*] Network packet filtering (replaces ipchains)  --->
          --- Network packet filtering (replaces ipchains)
          [ ]   Network packet filtering debugging
          [*]   Bridged IP/ARP packets filtering
          <M>   Netfilter netlink interface

                IP: Netfilter Configuration  --->
                    <M> Connection tracking (required for masq/NAT)
                    [*]   Connection tracking flow accounting
                    [*]   Connection mark tracking support
                    [ ]   Connection tracking events
                    <M>   Connection tracking netlink interface
                    <M>   FTP protocol support
                    <M>   IRC protocol support
                    <M>   TFTP protocol support
                    <M>   Amanda backup protocol support
                    <M>   PPTP protocol support
                    <M> IP Userspace queueing via NETLINK (OBSOLETE)
                    <M> IP tables support (required for filtering/masq/NAT)
                    <M>   limit match support
                    <M>   IP range match support
                    <M>   MAC address match support
                    <M>   Packet type match support
                    <M>   netfilter MARK match support
                    <M>   Multiple port match support
                    <M>   TOS match support
                    <M>   recent match support
                    <M>   ECN match support
                    <M>   DSCP match support
                    <M>   AH/ESP match support
                    <M>   LENGTH match support
                    <M>   TTL match support
                    <M>   tcpmss match support
                    <M>   Helper match support
                    <M>   Connection state match support
                    <M>   Connection tracking match support
                    <M>   Owner match support
                    <M>   Physdev match support
                    <M>   address type match support
                    <M>   realm match support
                    <M>   SCTP protocol match support
                    <M>   DCCP protocol match support
```

```
                              <M>   comment match support
                              <M>   Connection mark match support
                              <M>   Connection byte/packet counter match support
                              <M>   hashlimit match support
                              <M>   string match support
                              <M>   Packet filtering
                              <M>     REJECT target support
                              <M>   LOG target support
                              <M>   ULOG target support (OBSOLETE)
                              <M>   TCPMSS target support
                              <M>   NFQUEUE Target Support
                              <M>   Full NAT
                              <M>     MASQUERADE target support
                              <M>     REDIRECT target support
                              <M>     NETMAP target support
                              <M>     SAME target support
                              <M>   Packet mangling
                              <M>     TOS target support
                              <M>     ECN target support
                              <M>     DSCP target support
                              <M>     MARK target support
                              <M>     CLASSIFY target support
                              <M>     TTL target support
                              <M>     CONNMARK target support
                              <M>   raw table support (required for NOTRACK/TRACE)
                              <M>     NOTRACK target support
                              <M> ARP tables support
                              <M>   ARP packet filtering
                              <M>   ARP payload mangling

                          Bridge: Netfilter Configuration  --->
                              # 这个项目内容也一样，全部编成模块!

          # 底下同样的，可能的话就编译成为模块啊!
          <M> 802.1d Ethernet Bridging
          <M> 802.1Q VLAN Support
          < > DECnet Support
          < > ANSI/IEEE 802.2 LLC type 2 Support
          <M> The IPX protocol
          [ ]   IPX: Full internal IPX network
          <M Appletalk protocol support
          [*]   Appletalk interfaces support
          <M>     Apple/Farallon LocalTalk PC support
```

```
          <M>     COPS LocalTalk PC support
          [*]       Dayna firmware support
          [*]       Tangent firmware support
          <M>     Appletalk-IP driver support
          [*]       IP to Appletalk-IP Encapsulation support
          [*]       Appletalk-IP to IP Decapsulation support
          [*] QoS and/or fair queueing  --->
          <M> Firewall based classifier
          <M> U32 classifier
          [*]   U32 classifier performance counters
          [*] classify input device (slows things u32/fw)
          [*] Use nfmark as a key in U32 classifier
          <M> Special RSVP classifier
          <M> Special RSVP classifier for IPv6
          [*] Extended Matches
          (32)  Stack size
          <M>   Simple packet data comparison
          <M>   Multi byte comparison
          <M>   U32 hashing key
          <M>   Metadata
          <M>   Textsearch
          [*] Traffic policing (needed for in/egress)
              Network testing  --->

# 底下则是一些特殊的网络设备，例如红外线啊、蓝牙啊！
# 如果不清楚的话，就使用模块吧！除非你真的知道不要该项目！
[ ]   Amateur Radio support  --->
<M>   IrDA (infrared) subsystem support  --->
<M>   Bluetooth subsystem support  --->
<M>   Generic IEEE 802.11 Networking Stack
[ ]     Enable full debugging output
<M>     IEEE 802.11 WEP encryption (802.1x)
<M>     IEEE 802.11i CCMP support
<M>     IEEE 802.11i TKIP encryption
```

在这个设定项目当中，很多东西其实我们在基础篇还没有讲到， 因为大部分的参数都与网络、防火墙有关！由于防火墙是在启动网络之后再设定即可， 所以绝大部分的内容都可以被编译成为模块，而且也建议您编成模块！ 有用到才载入到核心即可啊！

- Device Drivers

这个是所有硬件装置的驱动程序库！哇！光是看到里面这么多内容，鸟哥头都昏了～ 不过，为了您自己的主机好，建议你还是得要一个项目一个项目的去挑选挑选才行～ 这里面的数据就与您主机的硬件有绝对的关系了！

```
Generic Driver Options  --->
          # 与韧体有关，保留默认值即可！

Connector - unified userspace <-> kernelspace linker  --->
Memory Technology Devices (MTD)  --->
          # 上面这两个不知道会不会用到的数据，暂时可以编译成为模块即可！

Parallel port support  --->
          <M> Parallel port support
          <M>   PC-style hardware
          <M>     Multi-IO cards (parallel and serial)
          <M>   Support for PCMCIA management for PC-style ports
          [*] IEEE 1284 transfer modes
          # 平行串行端口，呵呵！与打印机相关性挺高的！编译成为模块即可！

Plug and Play support  --->
          [*] Plug and Play support
          # 不啰唆，这个当然要选择啊！

Block devices  --->
          # 这里面与储存装置有关，全部给他编成模块！当然，确定不需要的，
          # 就不要编译咯！

ATA/ATAPI/MFM/RLL support  --->
          # 底下的设定你可以保留默认值，来让核心支持较为完整！
          # 不过，既然我们已经知道主机的硬件与主机板的芯片，当然可以作一些选择啰！
          <*> ATA/ATAPI/MFM/RLL support
          <*>   Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
          ---     Please see Documentation/ide.txt for help/info on IDE drives
          [ ]     Support for SATA (deprecated; conflicts with libata SATA driver)
                    # 这个不要选！因为 SATA 的新的驱动程序是包含在 SCSI 中的！
          [ ]     Use old disk-only driver on primary interface
          <*>     Include IDE/ATA-2 DISK support
          [*]     Use multi-mode by default
          <M>     PCMCIA IDE support
          <*>     Include IDE/ATAPI CDROM support
          <*>     Include IDE/ATAPI FLOPPY support
                    # 上面这两个就必选！让核心主动支持 CDROM 与软盘！

          <M>     SCSI emulation support
          [ ]     IDE Taskfile Access
          ---     IDE chipset support/bugfixes
          <*>     generic/default IDE chipset support
```

```
[*]     CMD640 chipset bugfix/support
[*]       CMD640 enhanced support
[*]     PNP EIDE support
          # 上面这几个也可以挑选起来，尤其会比较适合 Pentium 的主机！

          # 底下这几个则主要与主机板的芯片组有关啊！
[*]     PCI IDE chipset support
[*]       Sharing PCI IDE interrupts support
[ ]       Boot off-board chipsets first support
<*>       Generic PCI IDE Chipset Support
<*>       RZ1000 chipset bugfix/support
[*]       Generic PCI bus-master DMA support
                    # 底下这几个请特别挑选一番！
[ ]         Force enable legacy 2.0.X HOSTS to use DMA
[*]         Use PCI DMA by default when available
[ ]           Enable DMA only for disks
< >         AEC62XX chipset support
< >         ALI M15x3 chipset support
[ ]           ALI M15x3 WDC support (DANGEROUS)
< >         AMD and nVidia IDE support
< >         ATI IXP chipset IDE support
< >         CMD64{3|6|8|9} chipset support
< >         Compaq Triflex IDE support
< >         CY82C693 chipset support
< >         Cyrix/National Semiconductor CS5530 MediaGX chipset support
< >         HPT34X chipset support
< >         HPT36X/37X chipset support
< >         National SCx200 chipset support
<*>         Intel PIIXn chipsets support
<*>         IT821X IDE support
< >         NS87415 chipset support
<*>         PROMISE PDC202{46|62|65|67} support
[ ]           Special UDMA Feature
<*>         PROMISE PDC202{68|69|70|71|75|76|77} support
[*]           Enable controller even if disabled by BIOS
< >         ServerWorks OSB4/CSB5/CSB6 chipsets support
< >         Silicon Image chipset support
< >         SiS5513 chipset support
< >         SLC90E66 chipset support
< >         Tekram TRM290 chipset support
< >         VIA82CXXX chipset support
[ ]     Other IDE chipset support
[ ]     IGNORE word93 Validation BITS
```

```
          # 因为我的是 Intel 芯片组的主机板，所以全部无关的我都没有选择。
          # 不过，为了您自己好～其实，上面绝大部分的数据都给他做成模块比较妥当！

SCSI device support  --->
# 不论你有没有 SCSI 装置，你都必须要启动 SCSI 的支持！理由有二：
# 1. 因为 USB 装置用的就是仿真 SCSI 啊！
# 2. 因为 SATA 的设定项目就在这里面！
          < > RAID Transport Class
          <M> SCSI device support
          [*]   legacy /proc/scsi/ support
          ---   SCSI support type (disk, tape, CD-ROM)
          <M>   SCSI disk support
          <M>   SCSI tape support
          <M>   SCSI OnStream SC-x0 tape support
          <M>   SCSI CDROM support
          [*]     Enable vendor-specific extensions (for SCSI CDROM)
          <M>   SCSI generic support
          <M>   SCSI media changer support
          ---   Some SCSI devices (e.g. CD jukebox) support multiple LUNs
          [*]   Probe all LUNs on each SCSI device
          [*]   Verbose SCSI error reporting (kernel size +=12K)
          [*]   SCSI logging facility
                SCSI Transport Attributes  --->
                SCSI low-level drivers  --->
                    # 在这个项目当中，都保留默认值即可不过，
                    # 如果你有 SATA 的硬盘，请确认底下已经编译起来了！
                    <M> Serial ATA (SATA) support
                    <M>   AHCI SATA support
                    <M>   ServerWorks Frodo / Apple K2 SATA support
                    <M>   Intel PIIX/ICH SATA support
                    <M>   Promise SATA TX2/TX4 support
                    <M>   Pacific Digital SATA QStor support
                    <M>   VIA SATA support
                    <M>   VITESSE VSC-7174 SATA support
                PCMCIA SCSI adapter support  --->

Old CD-ROM drivers (not SCSI, not IDE)  --->
Multi-device support (RAID and LVM)  --->
          # 还记得不久之前才谈过的 LVM 吧？这里当然要选择啰！
          [*] Multiple devices driver support (RAID and LVM)
          <*>   RAID support
          <M>     Linear (append) mode
          <M>     RAID-0 (striping) mode
```

```
        <M>     RAID-1 (mirroring) mode
        <M>     RAID-4/RAID-5 mode
        <M>     RAID-6 mode
        <M>     Multipath I/O support
        <M>     Faulty test module for MD
        <M>   Device mapper support

Fusion device support  --->
IEEE 1394 (FireWire) support  --->
I2O device support  --->
        # 上面也编译成为模块即可！那个 IEEE 1394 就是我们常听到的『火线』。

Network device support  --->
        # 您总是有网络卡吧？所以啰～这里得要选择一个网络卡装置啊！
        [*] Network device support
        <M>   Dummy net driver support
        <M>   Bonding driver support
        <M>   EQL (serial line load balancing) support
        <M>   Universal TUN/TAP device driver support
        <M>   General Instruments Surfboard 1000
              ARCnet devices  --->
              PHY device support  --->
              Ethernet (10 or 100Mbit)  --->
                  # 这里面含有的就是 10/100 的网络卡！大部分都可以编成模块。
                  <M> RealTek RTL-8129/8130/8139 PCI Fast Ethernet Adapter support
                  <M> VIA Rhine support
                  # 上面这两个就是有名的螃蟹卡与 D-Link 530 所用的驱动程序。
              Ethernet (1000 Mbit)  --->
                  # 这里面含有的就是 10/100/1000 的网络卡！大部分都可以编成模块。
              Ethernet (10000 Mbit)  --->
              Token Ring devices  --->
              Wireless LAN (non-hamradio)  --->
              PCMCIA network device support  --->
              Wan interfaces  --->
        [*]   FDDI driver support
        < >     Digital DEFEA and DEFPA adapter support
        <M>     SysKonnect FDDI PCI support
        <M>   PLIP (parallel port) support
        <*>   PPP (point-to-point protocol) support
        [*]     PPP filtering
        <M>     PPP support for async serial ports
        <M>     PPP support for sync tty ports
        <M>     PPP Deflate compression
```

```
          # 如果您有 ADSL 拨接的话，呵呵！PPP 的装置也要选择上喔！
          < >     PPP BSD-Compress compression
          <M>   SLIP (serial line) support
          [*]     CSLIP compressed headers
          [*]     Keepalive and linefill
          [ ]     Six bit SLIP encapsulation
          [*]   Fibre Channel driver support

ISDN subsystem  --->
Telephony Support  --->
          # 上面这两个我都没有，所以并没有选择！

Input device support  --->
          # 这里面含有鼠标、键盘、游戏杆等等的输入装置，也是需要挑选的！
          --- Generic input layer (needed for keyboard, mouse, ...)
          ---   Userland interfaces
          ---   Mouse interface
          # 底下这三个与鼠标有关啦！也可以选择的！
          [ ]     Provide legacy /dev/psaux device
          (1024)  Horizontal screen resolution
          (768)   Vertical screen resolution
          <M>   Joystick interface
          < >   Touchscreen interface
          <*>   Event interface
          < >   Event debugging
          ---   Input Device Drivers
          ---   Keyboards  --->
          [*]   Mouse  --->
          [ ]   Joysticks  --->
          [ ]   Touchscreens  --->
                    # 我没有游戏杆也没有触控式面版，所以上面两个不选！
          [*]   Miscellaneous devices  --->
              Hardware I/O ports  --->

Character devices  --->
          # 里面的资料也很多，也要注意 AGP 显示卡的芯片组啊！
          # 而因为鸟哥的环境是 Intel 的芯片，所以自然将那个编进去，
          # 其它的成为模块即可！其它的除非确定不需要，否则保留默认值即可！
          <*> /dev/agpgart (AGP Support)
          <M>   ALI chipset support
          <M>   ATI chipset support
          <M>   AMD Irongate, 761, and 762 chipset support
          <M>   AMD Opteron/Athlon64 on-CPU GART support
```

```
          <*>   Intel 440LX/BX/GX, I8xx and E7x05 chipset support
          <M>   NVIDIA nForce/nForce2 chipset support
          <M>   SiS chipset support
          <M>   Serverworks LE/HE chipset support
          <M>   VIA chipset support
          <M>   Transmeta Efficeon support
          <M> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)
          <M>   3dfx Banshee/Voodoo3+
          <M>   ATI Rage 128
          <M>   ATI Radeon
          <M>   Intel I810
          <M>   Intel 830M, 845G, 852GM, 855GM, 865G
          <M>     i830 driver
          <M>     i915 driver
          <M>   Matrox g200/g400
          <M>   SiS video cards
          <M>   Via unichrome video cards
          < >   Savage video cards

I2C support  --->
          # 还记得我们去侦测主机板的温度与压力吧？呵呵！那就是透过核心的
          # 这个 I2C 的模块功能了！预设情况下，这个项目都有支持，所以，
          # 保留默认值即可。

Dallas's 1-wire bus  --->
          # 这个与某些热感应装置有关，可以不编译，也可以保留编成模块即可！

Hardware Monitoring support  --->
          # 这个也与 I2C 有点关系，他主要可以接受硬件的侦测，
          # 所以在这个项目内您会看到 LM_XX 之类的模块！啊！就保留模块即可！

Misc devices  --->
Multimedia Capabilities Port drivers  --->
Multimedia devices  --->
          # 类似影像撷取卡、FM 广播声卡等等，可在这里设定！
          # 如果您的主机是用作服务器，那么这里或许可以不要选择。
          # 当然啦，这个项目几乎都是模块，保留默认值也不错！

Graphics support  --->
          # 嘿嘿！重点之一，显示卡的芯片组～刚刚前面提到的都是主机板的
          # 对显示卡的总线支持 (PCI-E 与 AGP) ，这里则是针对显示卡芯片！
          # 鸟哥的显示卡是 NVidia 的，所以将他选择即可！其它的可以编成模块！
          <*> Support for frame buffer devices
```

```
---   Enable Video Mode Handling Helpers
---   Enable Tile Blitting Support
<M>   Cirrus Logic support
< >   Permedia2 support
< >   CyberPro 2000/2010/5000 support
< >   Arc Monochrome LCD board support
[ ]   Chips 69000 display support
[ ]   IMS Twin Turbo display support
<M>   VGA 16-color graphics support
[*]   VESA VGA graphics support
< >   Hercules mono graphics support
<M>   nVidia Framebuffer Support
<M>   nVidia Riva support
[ ]     Enable DDC Support
[ ]     Lots of debug output from Riva(nVidia) driver
<M>   Matrox acceleration
[*]     Millennium I/II support
[*]     Mystique support
[*]     G100/G200/G400/G450/G550 support
<M>     Matrox I2C support
<M>       G400 second head support
[*]     Multihead support
< >   ATI Radeon display support (Old driver)
<M>   ATI Radeon display support
[*]     DDC/I2C for ATI Radeon support
[ ]     Lots of debug output from Radeon driver
<M>   ATI Rage128 display support
<M>   ATI Mach64 display support
[*]     Mach64 CT/VT/GT/LT (incl. 3D RAGE) support
[*]       Mach64 generic LCD support (EXPERIMENTAL)
[ ]       Rage XL No-BIOS Init support
[*]     Mach64 GX support
<M>   SiS/XGI display support
<M>   NeoMagic display support
<M>   IMG Kyro support
<M>   3Dfx Banshee/Voodoo3 display support
<M>   3Dfx Voodoo Graphics (sst1) support
< >   Cyberblade/i1 support
<M>   Trident support
< > Epson S1D13XXX framebuffer support
< > Virtual Frame Buffer support (ONLY FOR TESTING!)
    Console display driver support  --->
    Logo configuration  --->
```

```
          [*] Backlight & LCD device support  --->

Sound  --->
          # 这个是声卡啊！鸟哥的机器上面没有声卡，所以直接不选。
          # 您可以进入后选择您的声卡啊！

USB support  --->
          # 这个则是 USB 的驱动模块！还记得我们在硬件维护的地方讲过的
          # USB 的模块名称吧？呵呵！在里面找找吧！
          # 其实这里面鸟哥仅保留默认值，然后再加上选择 USB 2.0 的支持，
          # 其它的 usb 装置全部给他勾成模块！这样就 OK 啦！

MMC/SD Card support  --->
          # 这是多媒体卡 (multi-media card) ，鸟哥是用不到的，所以不选！

InfiniBand support  --->
SN Devices  --->
          # 这两个应该也用不到，所以保留默认值即可！
```

在这里面真的很重要，因为很多数据都与你的硬件有关。核心推出时的默认值是比较符合一般状态的， 所以很多数据其实保留默认值就可以编的很不错了！不过，也因为较符合一般状态， 所以核心额外的编译进来很多跟你的主机系统不符合的数据，例如网络卡装置～ 你可以针对你的主机板与相关硬件来进行编译。不过，还是要记得有『未来扩充性』的考虑！ 之前鸟哥不是谈过吗，我的网络卡由螃蟹卡换成 3Com 时，核心捉不到～ 因为......我并没有将 3Com 的网络卡编译成为模块啊！ @_@

- File systems

档案系统的支持也是很重要的一项核心功能！因为如果不支持某个档案系统， 那么我们的 Linux kernel 就无法认识，当然也就无法使用啦！例如 Quota, NTFS 等等特殊的 filesystem 。底下是详细的资料啰！

```
<*> Second extended fs support
[*]   Ext2 extended attributes
[*]     Ext2 POSIX Access Control Lists
[*]     Ext2 Security Labels
[ ]   Ext2 execute in place support
<*> Ext3 journalling file system support
[*]   Ext3 extended attributes
[*]     Ext3 POSIX Access Control Lists
[*]     Ext3 Security Labels
[ ] JBD (ext3) debugging support
# EXT2/EXT3 是必选的吧！将他选择起来先！

<M> Reiserfs support
[ ]   Enable reiserfs debug mode
[*]   Stats in /proc/fs/reiserfs
```

```
[*]   ReiserFS extended attributes
[*]     ReiserFS POSIX Access Control Lists
[*]     ReiserFS Security Labels
<M> JFS filesystem support
[*]   JFS POSIX Access Control Lists
[*]   JFS Security Labels
[ ]   JFS debugging
[ ]   JFS statistics
<M> XFS filesystem support
[*]   XFS Quota support
[*]   XFS Security Label support
[*]   XFS POSIX ACL support
<M> Minix fs support
# 上面这几个 filesystem 不知道什么时候会用到，当然是编成模块比较好！

<M> ROM file system support
[*] Inotify file change notification support
[*] Quota support
< >   Old quota format support
<*>   Quota format v2 support
# Quota 够重要吧！务必要将他圈选起来才行喔！

<M> Kernel automounter support
<M> Kernel automounter version 4 support (also supports v3)
< > Filesystem in Userspace support
    CD-ROM/DVD Filesystems  --->
          <*> ISO 9660 CDROM file system support
          [*]   Microsoft Joliet CDROM extensions
          [*]   Transparent decompression extension
          <M> UDF file system support
          # 注意！那个 ISO 9660 的 filesystem 务必要挑选！

    DOS/FAT/NT Filesystems  --->
          <M> MSDOS fs support
          <M> VFAT (Windows-95) fs support
          (950) Default codepage for FAT
          (big5) Default iocharset for FAT
          <M> NTFS file system support
          [ ]   NTFS debugging support (NEW)
          [*]   NTFS write support
          # 哇！不但可以选择预设是中文语系，而且，还可以支持 NTFS
          # 可擦写哩！这一版真强！不过，NTFS 能否真的可以写入，不确定～
```

```
Pseudo filesystems  --->
        [*] /proc file system support
        [*]   /proc/kcore support
        [*] Virtual memory file system support (former shm fs)
        [*] HugeTLB file system support
        < > Relayfs file system support
        # 这几个是一定要的啦!

Miscellaneous filesystems  --->
        # 这里面的数据可以选择预设即可!

Network File Systems  --->
        <M> NFS file system support
        [*]   Provide NFSv3 client support
        [*]     Provide client support for the NFSv3 ACL protocol extension
        <M> NFS server support
        [*]   Provide NFSv3 server support
        [*]     Provide server support for the NFSv3 ACL protocol extension
        [*]   Provide NFS server over TCP support
        <M> SMB file system support (to mount Windows shares etc.)
        [*]   Use a default NLS
        (cp950) Default Remote NLS Option
        # 这里可以加上这个预设参数，支持中文语系啊!
        <M> CIFS support (advanced network filesystem for Samba, Window and othe
        [ ]   CIFS statistics
        [*]   CIFS extended attributes (EXPERIMENTAL)
        [*]     CIFS POSIX Extensions (EXPERIMENTAL)
        [ ]   CIFS Experimental Features (EXPERIMENTAL)
        <M> NCP file system support (to mount NetWare volumes)
        [*]   Packet signatures
        [*]   Proprietary file locking
        [*]   Clear remove/delete inhibit when needed
        [*]   Use NFS namespace if available
        [*]   Use LONG (OS/2) namespace if available
        [*]   Lowercase DOS filenames
        [*]   Use Native Language Support
        [*]   Enable symbolic links and execute flags
        <M> Coda file system support (advanced network fs)
        [ ]   Use 96-bit Coda file identifiers
        # 其实大部分仍然是模块的项目啦!

Partition Types  --->
        # 里面含有 Minix, sun 等等的磁盘分割表的格式支持,
```

```
        # 您如果确定不需要，可以将他拿掉就是了！

    Native Language Support  --->
        (utf8) Default NLS Option
        <*>   Traditional Chinese charset (Big5)
        # 其它都保留默认值即可，这两个项目稍微确认一下！
```

这部份也是有够麻烦～因为涉及核心是否能够支持某些档案系统，以及某些操作系统支持的 partition table 的支持项目。在进行选择时，也务必要特别的小心在意喔！ 尤其是我们常常用到的网络操作系统 (NFS/Samba 等等)，以及基础篇谈到的 Quota 等， 您都得要勾选啊！否则是无法被支持的。比较有趣的是 NTFS 在这一版的核心里面竟然有支持可写入的项目， 着实让鸟哥吓了一跳了！^_^

- Security options

这一部份与安全性比较有关。几乎保留默认值即可，仔细注意一下 SELinux 的项目， 该项目是美国国家安全局发展的 Linux 细部安全维护控件目，需要勾选才行！

- Cryptographic options

这部份则是加密参数的设定。一般我们使用的账号密码登入，利用的就是 MD5 这个加密机制，要让核心有支持才行啊！几乎所有的项目都给他做成模块即可！ 不过 MD5 与 SHA1 必须要直接由核心支持比较好！

```
--- Cryptographic API
---   HMAC support
<M>   Null algorithms
<M>   MD4 digest algorithm
<*>   MD5 digest algorithm
<*>   SHA1 digest algorithm
<M>   SHA256 digest algorithm
<M>   SHA384 and SHA512 digest algorithms
<M>   Whirlpool digest algorithms
<M>   Tiger digest algorithms
<M>   DES and Triple DES EDE cipher algorithms
<M>   Blowfish cipher algorithm
<M>   Twofish cipher algorithm
<M>   Serpent cipher algorithm
<M>   AES cipher algorithms (i586)
<M>   CAST5 (CAST-128) cipher algorithm
<M>   CAST6 (CAST-256) cipher algorithm
<M>   TEA, XTEA and XETA cipher algorithms
<M>   ARC4 cipher algorithm
<M>   Khazad cipher algorithm
<M>   Anubis cipher algorithm
<M>   Deflate compression algorithm
<M>   Michael MIC keyed digest algorithm
```

```
<M>   CRC32c CRC algorithm
```

- 取用旧数据与储存设定

还有底下这两个项目：

```
Load an Alternate Configuration File
Save Configuration to an Alternate File
```

这两个项目分别是储存刚刚做好的所有项目的设定数据，另一个则是将来自其它人作的选择给他读入！ 事实上，刚刚我们所做的设定只要在离开时选择 SAVE ，那么这些项目 通通会记录到目前这个目录下的 .config 档案内。 而我们也可以使用上面提到的 Save Configuration 这个项目来将刚刚做完的设定储存成另外的档案， 做成这个档案的好处是，你可以在下次在其它版本的核心作选择时，直接以 Load 来将这个档案的设定项目读入，这样可以减少您还要重新挑选一遍的困境啊！

在最初的画面上面选择 <Exit> 项目后，画面会出现一个询问你是否要储存的窗口，选择 Yes 后， 您所有的选择数据就都会被纪录到 .config 这个隐藏档案里面去了！有兴趣的话， 您可以使用 vi 去到该档案查阅一下，就知道你做过哪些设定啰！ ^_^

要请您注意的是，上面的资料主要是适用在鸟哥的个人机器上面的， 目前鸟哥比较习惯使用原本 distributions 提供的预设核心，因为他们也会主动的进行更新， 所以鸟哥就懒的自己重编核心了～ ^_^

此外，因为鸟哥重视的地方在于『网络服务器』上面，所以里头的设定少掉了相当多的个人桌上型 Linux 的硬件编译！所以，如果你想要编译出一个适合您的机器的核心， 那么可能还有相当多的地方需要来修正的！不论如何，请随时以 Help 那个选项来看一看内容吧！反正 Kernel 重编的机率不大！花多一点时间重新编译一次！ 然后将该编译完成的参数档案储存下来，未来就可以直接将该档案叫出来读入了！ 所以花多一点时间安装一次就好！那也是相当值得的！

核心的编译与安装

做完核心项目的选择啰～接下来呢？当然是编译与安装啦！核心的编译与安装很简单啦！来看看吧！

编译的流程

整个编译的过程真的很简单！作这个动作即可：

```
[root@linux linux-2.6.14.2]# make clean
# 将以前曾经进行过的 *.o 档案删除掉，这样比较不会产生新旧版本的误差！

[root@linux linux-2.6.14.2]# make bzImage
# 制作出核心档案！这个重要！过程很长啊！而且那个是大写的 I 喔！

[root@linux linux-2.6.14.2]# make modules
# 制作出模块相关的档案！
```

只要这三个动作，您的核心与模块就通通制作出来了！不过，制作出来的数据还是被放置在 /usr/src/linux-2.6.14.2 这个目录下～并没有被放到系统的相关路径中喔！ 在上面的过程当中，如果有发生任何错误的话，那么很可能是核心项目的挑选选择的不好， 可能您需要重新以 make menuconfig 再次的检查一下您的相关设定喔！ 如果还是无法成功的话，那么或许将原本的核心数据内的 .config 档案，复制到您的核心原始文件目录下， 然后据以修改，应该就可以顺利的编译出您的核心了。注意到，下达了 make bzImage 后， 最后的结果应该会像这样：

```
Root device is (3, 2)
Boot sector 512 bytes.
Setup is 7016 bytes.
System is 1721 kB
Kernel: arch/i386/boot/bzImage is ready  (#1)
[root@linux linux-2.6.14.2]# ll arch/i386/boot/bzImage
-rw-r--r--  1 root root 1770185 Dec  2 14:32 arch/i386/boot/bzImage
```

如此一来，您就可以发现您的核心已经编译好而且放置在 /usr/src/linux-2.6.14.2/arch/i386/boot/bzImage 里面啰～那个就是我们的核心档案！最重要就是他啦！我们等一下就会安装到这个档案哩！ 然后就是编译模块的部分啰～ make modules 进行完毕后，就等着安装啦！ ^_^

模块安装时的注意事项：

要强调的还是得强调，是这样的，在上面的介绍里，我们不是说过 Kernel 的外挂模块是放在 /lib/modules/`uname -r` 吗？好了，那么现在来想一想，如果你的『同一版本的核心编译两次』 的情况下，会怎样？这是很可能的情况呢！怎么说？万一你的第一次的编译没有成功的话，那总得编译第二次吧？而由于第一次编译完成之后，你的一些模块已经放在 /lib/modules/2.6.14.2 当中了(以这一次我们使用的核心版本为例)，那么下次在编译完成后，核心的模块还是会放在 /lib/modules/2.6.14.2 这个目录下，那不是重复了吗？有些模块会被重复放置，导致问题重重的～ 因此上，如果同一个版本的核心被编译两次以上的话，那么请将 /lib/modules 里面的该版核心先移动掉吧！举个例子来说，假如你的核心版本是 2.6.14.2 ，而又要对 2.6.14.2 重新编译一次，那么就必需要：

```
[root@linux ~]# cd /lib/modules
[root@linux modules]# mv 2.6.14.2 2.6.14.2.old
```

这样才行呢！不然安装之后还是会有问题的呦！请特别留意呢！

处理完毕后，开始要安装模块了～安装的方法很简单，直接这样做就好了！

```
[root@linux linux-2.6.14.2]# make modules_install
```

整个模块就安装到 /lib/modules 里面去喔～一般来说，目录名称会是 /lib/modules/2.6.14-2 ，但是如果您有填写核心附版本的话， 就会出现类似 2.6.14-1.1644_FC4 之类的目录名称啰！ ^_^ 接下来，就是准备要进行核心的安装了！哈哈！又跟 grub 有关啰～

安装旧版与新版的核心成多重开机系统

在编译好核心之后，我们已经知道核心档案放置在 /usr/src/linux-2.6.14.2/arch/i386/boot/bzImage ，而我们也晓得一部主机是可以做成多重开机系统的！这样说，应该知道鸟哥想要干嘛了吧？ 呵呵！对啦！我们将同时保留旧版的核心，并且新增新版的核心在我们的主机上面。

这样做有什么好处呢？最大的好处是可以确保能够顺利开机啦！因为核心虽然被编译成功了， 但是并不保证我们刚刚挑选的核心项目完全适合于目前这部主机系统， 可能有某些地方我们忘记选择了，这个将导致新核心无法顺利驱动整个主机系统，更差的情况是， 您的主机无法成功开机完成！此时，如果我们保留旧的核心， 呵呵！若新核心测试不通过，就用旧核心来启动啊！嘿嘿！保证比较不会有问题嘛！

关于多重开机的设定详情请参考 开机关机流程与 Loader 那一章，我这里不详细的说明了。我只假设您与我一样使用 grub 开机管理程序， 那么只要这样做，就能够设定好您的新核心了！

```
1. 移动新核心到 /boot 里面去:
[root@linux ~]# cp /usr/src/linux-2.6.14.2/arch/i386/boot/bzImage \
> /boot/vmlinuz-2.6.14-2
# 就一般的习惯而言，鸟哥建议您将核心档名设定成以 vmlinuz 为首的名称，
# 比较容易管理啦！
[root@linux ~]# cp /usr/src/linux-2.6.14.2/System.map \
> /boot/System.map-2.6.14-2

2. 修改 grub 设定档
[root@linux ~]# vi /boot/grub/menu.lst
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title VBird linux 2.6.14-2
        root (hd0,0)
        kernel /vmlinuz-2.6.14-2 ro root=/dev/hda2 rhgb quiet vga=788
title Fedora Core (2.6.11-1.1369_FC4)
        root (hd0,0)
        kernel /vmlinuz-2.6.11-1.1369_FC4 ro root=/dev/hda2 rhgb quiet vga=788
        initrd /initrd-2.6.11-1.1369_FC4.img
# 这个是鸟哥我的个人环境，您请依照您的主机系统来编写这个档案！
```

嘿嘿！这样才算成功的完成了整个核心的编译与安装～接下来呢？ 当然就是 reboot 去测试一下新核心是否可以顺利的启动您的系统啦！ 加油的啦！ ^_^

额外(单一)模块编译：

我们现在知道核心所支持的功能当中，有直接编译到核心内部的，也有使用外挂模块的，外挂模块可以简单的想成 就是驱动程序 啦！那么也知道这些核心模块依据不同的版本， 被分别放置到 /lib/modules/`uname -r`/ 目录中，各个硬件的驱动程序则是放置到 /lib/modules/`uname -r`/kernel/drivers/ 当中！而这些模块与装置代号的对应， 就必须要被写入 /etc/modprobe.conf 档案当中了。更多与 modprobe.conf 的数据请参考 开机流程与 loader 章节啰！

另外，关于模块的管理方面，我们也已经在 开机流程与 loader 当中稍微提过了。事实上，我们的 Linux 核心真的是越来越聪明了， 一般来说，当我们的软件有使用到核心的某项功能时，其实核心是会『主动的』去加载该功能的！ 根本不需要使用什么 modprobe 还是 insmod 之类的指令去加载呢！不过，有时候某些程序写的不好时， 确实可能需要我们手动来加载模块就是了。

那么在 Linux kernel 2.6 版里面的模块文件名是怎样呢？这个得要特别说明一下啰。在 kernel 2.4 版以前，模块的文件名都是 *.o 的，例如 vfat.o 这个档案系统模块就放在：

- /lib/modules/`uname -r`/kernel/fs/vfat/vfat.o

但是在 kernel 2.6 版以后，所有的核心模块都被改名字成为 *.ko 了！所以，如果你有 vfat 的模块， 他就会被放置到：

- /lib/modules/`uname -r`/kernel/fs/vfat/vfat.ko

请特别留意这个差异喔！ ^_^。此外，由于我们的核心原本就有提供很多的核心工具给硬件开发商来使用，而硬件开发商也需要针对核心所提供的功能来设计他们的驱动程序模块，因此， 我们如果想要自行使用硬件开发商所提供的模块来进行编译时，就需要使用到核心所提供的原始档当中， 所谓的标头档案 (header include file) 来取得驱动模块所需要的一些函式库或标头的定义啦！ 也因此我们常常会发现到，如果想要自行编译核心模块时，就得要拥有核心原始码嘛！

那核心原始码我们知道他是可能放置在 /usr/src/ 底下，早期的核心原始码被要求一定要放置到 /usr/src/linux/ 目录下，不过，如果您有多个核心在一个 Linux 系统当中，而且使用的原始码并不相同时， 呵呵～问题可就大了！所以，在 2.6 版以后，核心使用比较有趣的方法来设计他的原始码放置目录，那就是以 /lib/modules/`uname -r`/build 及 /lib/modules/`uname -r`/source 这两个连结档来指向正确的核心原始码放置目录。如果以我们刚刚由 kernel 2.6.14.2 建立的核心模块来说， 那么他的核心模块目录底下有什么咚咚？

```
[root@linux ~]# ls -l /lib/modules/2.6.14.2/
lrwxrwxrwx  1 root root     23 Dec  2 15:45 build -> /usr/src/linux-2.6.14.2
drwxr-xr-x  9 root root   4096 Dec  2 15:46 kernel
-rw-r--r--  1 root root 216725 Dec  2 15:46 modules.alias
-rw-r--r--  1 root root     69 Dec  2 15:46 modules.ccwmap
-rw-r--r--  1 root root 176206 Dec  2 15:46 modules.dep
-rw-r--r--  1 root root    739 Dec  2 15:46 modules.ieee1394map
-rw-r--r--  1 root root    206 Dec  2 15:46 modules.inputmap
-rw-r--r--  1 root root  16383 Dec  2 15:46 modules.isapnpmap
-rw-r--r--  1 root root 175001 Dec  2 15:46 modules.pcimap
-rw-r--r--  1 root root  83299 Dec  2 15:46 modules.symbols
-rw-r--r--  1 root root 231507 Dec  2 15:46 modules.usbmap
lrwxrwxrwx  1 root root     23 Dec  2 15:45 source -> /usr/src/linux-2.6.14.2
```

其中比较有趣的除了那两个连结档之外，还有那个 modules.dep 档案也挺有趣的， 那个档案是记录了核心模块的相依属性的地方，依据该档案，我们可以简单的使用 modprobe 这个指令来加载模块呢！至于核

心原始码提供的标头档，在上面的案例当中， 则是放置到 /usr/src/linux-2.6.14.2/include/ 目录中，当然就是藉由 build/source 这两个连结档案来取得目录所在的啦！^_^

单一模块编译

想象两个情况：

- 如果我的预设核心忘记加入某个功能，而且该功能可以编译成为模块，不过， 预设核心却也没有将该项功能编译成为模块，害我不能使用时，该如何是好？
- 如果 Linux 核心原始码并没有某个硬件的驱动程序 (module) ，但是开发该硬件的厂商有提供给 Linux 使用的驱动程序原始码，那么我又该如何将该项功能编进核心模块呢？

很有趣对吧！不过，在这样的情况下其实没有什么好说的，反正就是 『去取得原始码后，重新编译成为系统可以加载的模块』啊！很简单，对吧！^_^ 但是，上面那两种情况的模块编译行为是不太一样的，不过，都是需要 make, gcc 以及核心所提供的 include 标头档与函式库等等。

- 硬件开发商提供的额外模块：

很多时候，可能由于核心预设的核心驱动模块所提供的功能您不满意， 或者是硬件开发商所提供的核心模块具有更强大的功能， 又或者该硬件是新的，所以预设的核心并没有该硬件的驱动模块时，那您只好自行由硬件开发商处取得驱动模块， 然后自行编译啰！

如果您的硬件开发商有提供驱动程序的话，那么真的很好解决，直接下载该原始码，重新编译， 将他放置到核心模块该放置的地方后，呵呵！就能够使用了！举例来说，如果您不想使用核心原本提供的 Intel 网络卡模块，而想使用 Intel 官方释出的最新模块，例如下面这个例子：

- 模块说明与下载：http://downloadfinder.intel.com/scripts-df-external/Detail_Desc.aspx?agr=Y&Inst=Yes&ProductID=993&DwnldID=2896&strOSs=39&OSFullName=Linux*&lang=eng

您可以利用各种方法将他下载后，假设这个档案放置到 /root ，那么直接将他解压缩吧！ 之后就可以读一读 INSTALL/README ，然后找一下 Makefile ，就能够编译了。整体流程有点像这样：

```
1. 将档案解压缩：
[root@linux ~]# cd /usr/local/src
[root@linux src]# tar -zxvf /root/e100-3.4.14.tar.gz
[root@linux src]# cd e100-3.4.14

2. 开始进行编译与安装：
[root@linux e100-3.4.14]# vi README  <==注意查一下该档案内容
[root@linux e100-3.4.14]# cd src
[root@linux src]# make
# 此时您会看到出现如下这一行：
# make[1]: Entering directory `/usr/src/kernels/2.6.13-1.1532_FC4-i686'
# 这代表这个驱动程序在编译时，会去读取的核心原始码 include file
```

```
# 的目录所在！有兴趣的朋友，务必查阅一下 Makefile 啦！
[root@linux src]# ls -l
-rw-r--r--  1 root root  77908 Jul  2 08:24 e100.c
-rw-r--r--  1 root root 351351 Dec  5 00:48 e100.ko
-rw-r--r--  1 root root   4775 Dec  5 00:48 e100.mod.c
-rw-r--r--  1 root root  39684 Dec  5 00:48 e100.mod.o
-rw-r--r--  1 root root 312564 Dec  5 00:48 e100.o
-rw-r--r--  1 root root  21092 Jul  2 08:24 ethtool.c
-rw-r--r--  1 root root  43258 Jul  2 08:24 kcompat.h
-rw-r--r--  1 root root   9610 Jul  2 08:24 Makefile

3. 开始将该模块移动到核心目录，并且更新模块相依属性！
[root@linux src]# cp e100.ko  \
> /lib/modules/`uname -r`/kernel/drivers/net
[root@linux src]# cd /lib/modules/`uname -r`
[root@linux 2.6.13-1.1532_FC4]# depmod -a
```

有趣吧！透过这样的动作，我们就可以轻易的将模块编译起来，并且还可以将他直接放置到核心模块目录中， 同时以 depmod 将模块建立相关性，未来就能够利用 modprobe 来直接取用啦！^_^ 但是需要提醒您的是，当自行编译模块时， 若您的核心有更新（例如利用自动更新机制进行在线更新）时，则您必须要重新编译该模块一次， 重复上面的步骤！才行！因为这个模块仅针对目前的核心来编译的啊！对吧！

- 利用旧有的核心原始码进行编译：

举个例子来说，鸟哥目前 FC4 的核心就是 2.6 版，而且也有 NTFS 的原始码，只不过， FC4 就是没有将这个模块给他编译起来！那我能否使用目前的核心原始码进行 NTFS 档案系统的模块编译呢？当然可以啊！不过，我是否需要整个核心编译的过程从头来一次呢？ 呵呵！当然不需要啊！否则～多麻烦～那该怎么作？

很简单啦～我们首先到目前的核心原始码所在目录下达 make menuconfig ， 然后将 NTFS 的选项设定成为模块，之后直接下达：

make fs/ntfs/

那么 ntfs 的模块就会自动的被编译出来了！可惜的是，预设的 FC4 核心原始码并没有附上所有的程序代码， 仅有提供相关的 Makefile 档案而已，伤脑筋～ 因此，您仅能以我们刚刚才建立的 /usr/src/linux-2.6.14.2 这个目录， 直接下达 make fs/ntfs 来建立起 ntfs.ko 这个模块～ 然后将该模块复制到 /lib/modules/2.6.14.2/kernel/fs/ntsf/ 目录下， 再去到 /lib/modules/2.6.14.2 底下执行 depmod -a ，呵呵～ 就可以在原来的核心底下新增某个想要加入的模块功能啰～ ^_^

核心模块管理： lsmod, modinfo, modprobe, insmod, rmmod...

核心与核心模块是分不开的，至于驱动程序模块在编译的时候，更与核心的原始码功能分不开～ 因此，您必须要先了解到：核心、核心模块、驱动程序模块、核心原始码与标头档案的相关性， 然后才有办法了解到为何编译驱动程序的时候老是需要找到核心的原始码才能够顺利编译！ 然后也才会知道，为何当核心更新之后，自己之前所编译的核心模块会失效～

此外，与核心模块有相关的，还有那个很常被使用的 modprobe 指令， 以及开机的时候会读取到的模块定义数据文件 /etc/modprobe.conf ， 这些数据您也必须要了解才行～相关的指令说明我们已经在开机流程与 loader 文章内谈过了， 您应该要自行前往了解喔！ ^_^

本章习题练习

(要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

一些基础的 Linux 问题

最近更新日期：2002/06/28

一些基础的 Linux 问题与讨论：

注意：如果您有更好的试题，或者是有相关的数据要提供给 VBird 的话，我也会尽快的将他写到网页中的！感谢感谢！！ ^_^ ..

一、什么是 Linux 呢？！

1. 试说明 Unix 与 Linux 的历史？
2. 简单说明 GNU General Public License （GPL）的精神：
3. 什么是 Linux Distribution?
4. 市面上众多的 Linux 版本当中，有何异同？
5. Linu Kernel 的功能：
6. 试说明 Linux Kernel 与使用者之间的相关性？
7. 试说明什么是 GUI
8. Linux 的优点：
9. Linux 是多人多任务的环境，请问何谓多任务(Multitask)与多人(Multiuser)：

二、Linux 的安装与认识 Linux 支持的硬件

1. 一般而言， Linux 需要的配备并不需要太高档，但 Open Linux Server 3.1.1 需要？
2. 什么是 IDE 界面，一般而言，普通 PC 允许几个 IDE 界面与装置？
3. IDE2 的 master 之第一个 logical 磁盘中，其装置代号（文件名称）为何？
4. 在硬盘分割 (Partition)时，最多有几个 primary + extended ？
5. 若在分割的时候，在 IDE1 的 slave 硬盘中，分割『六个有用』的扇区（具有 filesystem 的），此外，有两个 primary 的扇区！请问六个扇区的代号？
6. 一般而言安装 Linux 至少要有哪两个 partition 呢？
7. 一般而言，在 RAM 为 64MB 或 128 MB 的系统中， swap 要开多大？
8. 光驱与软盘机的代号（装置文件名称）？
9. 网络卡（Network Interface Card, NIC）的代号？
10. 预设的 Linux 档案格式为何？又，目前常用的 Journalling 档案格式为何？
11. 安装 Linux 的主要流程？
12. 什么是 GMT 时间？台北时间差几个钟头？
13. Tap, SCSI 硬盘, RAID, printer 的装置代号？

其它注意事项：

- 要玩 X-window 时，至少需要具有 4-8 MB 的 VGA RAM 才好！
- 第一次使用与安装 Linux 时，请尽量的安装『所有套件！』
- 若具有安装与使用经验之后，则安装的时候，仅选择您所预计需要开放的服务之套件即可！
- 特别留意硬件对于 Linux distribution 的支持度！可以参考一下如下的网页：http://www.linux.org.tw/hardware/index.php3

- 制作 Install 之软盘开机片：
  将可开机光盘放入 Windows 系统下，并执行光驱中的档案(假设光盘代号为 E:)

```
E:\col\tools\rawrite\rawrite2
Enter disk image source filename:
E:\col\launch\floppy\install.144
Enter target disketie derive: A:
```

- 制作软盘开机片的一般方法：
  mkbootdisk --device /dev/fd0 `uname -r`

三、简易 KDE 的使用(X-Window)

1. 若以 X-Window 为预设的登入方式，那请问如何进入 Virtual console 呢？
2. 如何修改进入 Linux 时候的 run-level ？又 run-level 0, 1, 3, 5 各代表什么意思？
3. XFree86 在 X-Window 中的角色为何？
4. XFree86 的主要设定档在哪里？
5. KDE 预设提供多少个 Virtual desktops ？

四、在线求助 (Help)

1. 在 Linux 系统中，安装的套件 (RPM安装) 预设的文件数据放在那个目录底下？
2. 查看 Linux 系统的在线求助可以使用什么指令？
3. 使用 man 来查询在线求助的时候，若要查询类似参数档(如/etc/passwd)需如何？
4. 使用 man 查询 command 的所有相关指令时，需要下什么参数？
5. man page 的 man path 变量，记录在那个档案中呢？！

五、vi 的使用

1. 在 linux 底下最常使用的文书编辑器为 vi ，请问如何进入编辑模式？
2. 如何由编辑模式跳回一般模式？
3. 若上下左右键无法使用时，请问如何在一般模式移动光标？
4. 若 [pagedown] [pageup] 在一般模式无法使用时，如何往前或往后翻一页？
5. 如何到本档案的最后一行、第一行；本行的第一个字符、最后一个字符？
6. 如何删除一行、n行；如何删除一个字符？
7. 如何复制一行、n行并加以贴上？
8. 如何搜寻 string 这个字符串？
9. 如何取代 word1 成为 word2，而若需要使用者确认机制，又该如何？
10. 如何读取一个档案 filename 进来目前这个档案？
11. 如何另存新档成为 newfilename？
12. 如何存档、离开、存档后离开、强制存档后离开？
13. 如何设定与取消行号？

六、档案与目录管理

1. 请问底下的目录与主要放置什么数据？
2. 『旧的』 Unix 系统与 Linux 系统的『文件名』最多可以容许几个字符？
3. 什么是绝对路径与相对路径，要由 /usr/share/doc 进入到 /usr/share/man 由相对路径与绝对路径的写法各为何？
4. 在非为根目录的任何一个目录中，下达 ls －al 时，均会有『.』及『..』这两个目录，请问分别代表什么？
5. 显示、变换目录的时候，使用什么指令？
6. 新增目录、移除目录移动目录与拷贝目录有什么指令可用？
7. 如何查看一个档案的『内容』（不要使用 vi 的情况下）
8. 什么是 hard link 与 soft link 的档案？有何不同？
9. 如何在 root 的家目录下建立一个 /bin 的连结快捷方式
10. 若有一个连结档为 testing 连结到 test1.sh ，请问 copy testing 到 /tmp 底下，『cp testing /tmp』则 /tmp/testing 这个档案会是：(1) 连结档，连结到 test1.sh ；(2) 一般档案，为内容与 test1.sh 相同？
11. 当一个档案属性为 －rwxrwxrwt 则表示这个档案的意义为？
12. 我需要将一个档案的属性改为 －rwxr-xr-- 请问该如何下达指令？
13. 在 /usr/bin/passwd 这个档案的属性为 －r-s--x--x 请问代表什么意思(s)
14. 如何改出 －rwxr-sr-x 与 －rwxr-xr-t这个属性？
15. 如何查看一个档案的类型 (type) 例如纯文字文件、执行文件与 setuid 档案等？
16. 若一个使用者的 umask 为 033 ，请问他建立一个新的目录与档案时的预设属性为？
17. 若我需要更改一个档案的拥有者与群组，该用什么指令？
18. 如何将一个档案的修改日期改成目前的时间？
19. 如何搜寻一个档案？
20. Linux 预设的档案系统为何？此外，常用的 Journalling 档案格式有哪些？
21. less 跟 more 有什么不同？
22. 在比较两个档案的异同时，常使用的是 diff 与 cmp ，请教两者有何不同？

六.1、磁盘挂载与虚拟内存问题

1. 我要如何查看 Linux 系统当中所有已经挂载的硬盘容量与 inodes ？
2. 我要如何查看目前所在目录的所有档案占用的硬盘空间，此外，如何仅输出结果？
3. 如果扇区 /dev/hda3 有问题，偏偏他是被挂载上的，请问我要如何修理此一扇区？
4. 承上题，那么有问题的档案将被移动到那个目录下？
5. 试说明新增一个 partition 在 /dev/hdb 当中，且为 hdb5 时，并挂载上 /disk2 ，需要哪些步骤？
6. 如何新增加 64MB 的虚拟内存 swap ？
7. 如果要设定一个新挂载上的扇区，令他可以在开机的时候被挂载上，应该编辑哪一个档案？
8. 简易说明 quota 的设置流程？

七、BASH SHELL

1. 简单说明 bash shell 的功能特征：

2. 在设定变量中，主要的规则为何？
3. 显示环境变量与所有变量的指令为何？
4. 在环境变量当中，『使用者家目录、主机名称』的变量名称为：
5. 如何让一个变量可以持续到下一个程序 (PID) 去？
6. 在变量的设定当中，双引号与单引号有何不同？
7. Command1 `command2` 这个指令代表什么意思？
8. 在命令重导向当中， > 与 >> 有什么不同？
9. 若要将错误的输出导向『不见了』的装置，该如何是好？
10. 在管线指令中，我要将 last 输出的结果显示 root 的『登入次数』， 如何做？
11. 如何设定一个变量名称为 name ，使其显示为『 VBird’s testing variable. 』？
12. 请问 bash shell 的环境变量与自己的个人变量可以登入便设定的档案？
13. 如何查询曾经操作过的指令？如何执行第 26 个操作过的指令？上一个操作过的指令？
14. 如何设定命令别名？使 lm 与 ls -al|more 功能相同
15. 如何将 /bin 的所有信息输出到 /home/testing.dat 这个档案？
16. 在上一题中，若还要输出到屏幕上，要如何是好？
17. 执行 find ，并将正确的结果输出到 right 而错误结果输出到 error 当中？
18. 如何在指令列模式中将 /root/.bashrc 的资料加在 /home/col/.bashrc 当中？
19. 不要显示执行的结果要如何是好？
20. 列出这个月曾经登入主机的使用者信息即可，不需要重复？
21. 万用字符当中， *, ?, [] 各代表什么意思？

七.1、SHELL SCRIPTS

1. 什么是 shell scripts ？ scripts 有何功用？
2. 要撰写可以经由键盘输入的 scripts 时，常使用到的指令？
3. 在 shell script 当中，在控制式中， && 与 || 代表什么意思？
4. 在使用循环的时候，常使用到的控制式为哪三个？
5. 执行 scripts 的方法有哪两种？
6. 如何宣告一个变量成为整数型态？
7. 为何在 shell script 前面都要宣告 shell 呢？
8. 在判断式中，如何判断一个档名是否存在？
9. 在执行一个 scripts 时，在 scripts 内的变量， $0, $1 代表什么？
10. 若要写一个既定的 parameter （如 start ），用哪一个判断式较简单？
11. 如何印出目前系统中的所有账号，并加以排序且输出到 /tmp/account 中？
12. 如何在不执行 scripts 的情况中 debug ？

八、基本账号管理

1. 如何新增一个使用者 username，且该使用者没有家目录？
2. Linux 使用者的账号、密码与群组的名称档案放在哪里？
3. 建立新使用者的预设家目录内容在那个目录中？
4. 建立一个新使用者时，其使用到的相关档案有哪些？
5. root 的 UID 与 GID 各为何？
6. 如何让一个使用者不能登入主机，但是可以收信？
7. 试说明一个使用者登入系统的流程？

8. 在 /etc/shadow 当中的日期设定中，其数字代表的意义为何？
9. 如何变更使用者的一些属性？
10. 如何视察一个使用者 username 所拥有的群组呢？
11. 试说明如何手动增加一个使用者 username 与群组 groupname？
12. 试说明 /etc/passwd 这个档案的内容与格式：
13. 使用 id 这个指令时，可以显示什么讯息数据？
14. 可以控制使用者使用主机资源的预设档案在哪里？
15. 基本的压缩指令有哪些？

九、开机程序

1. 试说明开机流程：
2. 开机时后的加载讯息可以看哪里？
3. 改变 run-level 或关机的指令
4. 改变登入 Linux 时候预设的 run-level 要改那个档案？

十、程序与资源管理

1. 如何查看目前的程序？
2. 如何查看目前的内存使用状况？
3. 目前的工作如何丢到背景中？
4. 如何取得目前背景中的工作，且将他拉回前景中？
5. 如何设定一个程序在开始执行时候的优先值？
6. 什么指令可以修改一个正在执行的程序之 nice 值？
7. 在例行性命令中，使用的两支 demane 是什么？
8. 使用 crontab 这个『指令』的时候，如何可以查看目前的工作与删除目前的工作？
9. 常用的 kill 指令之 signal 当中， 1, 9, 15 代表什么？

来看看解答啰：
请注意：这些解答是 VBird 自己查书或者是实际操作所得到的答案，如果您发现这些答案是『错误的』请赶快跟 VBird 联络，好让我将数据赶快的订正！感谢大家的热情支持啰！

一、什么是 Linux 呢？！

- 试说明 Unix 与 Linux 的历史？
    - Multics 系统：由 Bell（贝尔实验室）、MIT（麻省理工学院）与 GE（美国通用电器）合作开发的一个系统；
    - 1969：K. Thompson 替 DEC 公司写了一个简单的 file system 系统，此为 Unix 的前身，但是 Unix 一词尚未出现
    - 1973：由 Bell 的 D. Richie 以 C 语言改写了 Thompson 的小系统，全部以 ASCII 档案进行改写，方便于应用！此时为第一次出现 Unix 这个操作系统的名词，不过，由于 Unix 乃针对不同的硬件而设定，因此仅出现在大型的 Server 上面看到！经过数年后，有底下几种版本：

1. System V 来自于 ATT 公司；
2. BSD 来自于加州理工学院；
3. AIX 来自于 IBM 公司

- 1979：Richard Stallman 倡导 Open source 精神；
- 1984：GNU 与 Free Software Foundation (FSF)由 R. Stallman 倡导；
- 1986：Xfree86 出现在 Unix 上面，且在 1994 年整合于 Linux 中！
- 1991：芬兰大学生 Linus Torvalds 在网络上首次公告 0.02 版的 Linux Kernel ，称为 hobby。

- 简单说明 GNU General Public License （GPL）的精神：
  - GPL 的授权之软件，乃为自由软件（Freeware），任何人皆可拥有他；
  - 开发 GPL 的团体(或商业企业)可以经由该软件的服务来取得服务的费用；
  - 经过 GPL 授权的软件，其属于 Open source 的情况，所以应该公布其原始码；
  - 任何人皆可修改经由 GPL 授权过的软件，使符合自己的需求；
  - 经过修改过后 Open source 应该回馈给 Linux 社群。

    (PS. Open source 最大的优点为多人维护， debugs 的速度较快，程序亦较为安全，但是缺点则是缺乏『专人』维护！）

- 什么是 Linux Distribution？

  基本上 Linux 是在 1991 年由芬兰大学生 Linus Torvalds 写的一个核心操作系统，最早仅只是核心而已，后来由于此一核心不但可以适合于主流 PC 的 x86 架构，并且稳定，因此有相当多的团队加入研究开发，后来某些公司将一些套件加入此一核心中，变成为完整的『安装光盘』，亦即是 Linux Distribution 了！所以才会有这么多的 Linux 版本，各版本之间没有所谓的谁优谁劣，而是个有其优缺点！看你适合哪一款，就用那一款吧！

  开发商针对 Linux Kernel 进行开发，并加入适合该 Kernel 的套件（如 ftp, apache, mail 等等），及配合开发商本身的支持软件，而制作出来的可安装光盘即可称为 distribution。

- 市面上众多的 Linux 版本当中，有何异同？

  唯一相同的地方在于 Linux 的『Kernel』，目前（2002/06/29）Kernel 发展至 2.4.xx 版本，至于不同点则是 Linux 开发商自行加入或者是研发的软件。

- Linu Kernel 的功能：
  - System call interface
  - Process control
  - Memory management
  - File System management
  - Device drivers

  简单的说，任何跟系统硬件资源有关的都是 Linux Kernel 管辖的范围，所以编辑核心的时候，将核心编的越小越好！

- 试说明 Linux Kernel 与使用者之间的相关性？
  硬件<=>Kernel (Modules Loader)<=>Shell (Terminal or GUI)<=>Users

- 试说明什么是 GUI
  GUI 为 Graphical User Interface 的简写，即为使用者图形界面，目前我们在 Linux 上面的 GUI 一般称为 X-Window ，而其核心为 Xfree86 这个 X-Server ！请注意，这个 Xfree86 最大的功能即是在控制『显示卡、硬件周边』等跟 Window 有关的界面，所以 X-Window 无法启动时，通常是 Xfree86 这个 X-Window 的核心驱动程序设定不完全有关！

- Linux 的优点：
  最大的优点来自于其良好的资源分配！所以具有：
    - 良好的多人多任务环境，资源分配平均；
    - 除了免费之外，在线更新速度快，除错与安全性均较佳；
    - 为 Open source 的授权，故而具有 open source 的所有优缺点；
    - 网络功能强大。

- Linux 是多人多任务的环境，请问何谓多任务(Multitask)与多人(Multiuser)：
    - Multitask 指的是多任务环境，在 Linux 系统下， CPU 与其它例如网络资源可以同时进行多项工作， Linux 最大的特色之一即在于其多任务时，资源分配较为平均！
    - Multiuser 指的是 Linux 允许多人同时连上主机之外，每个使用者皆有其各人的使用环境，并且可以同时使用系统的资源！

二、Linux 的安装与认识 Linux 支持的硬件

- 一般而言， Linux 需要的配备并不需要太高档，但 Open Linux Server 3.1.1 需要？
    - CPU 需要在 PII, PIII, P4 或 K7, K8 以上等级；
    - RAM 至少需要 64 MB，
    - 硬盘至少 550 MB，全部安装则需要 1.7GB

- 什么是 IDE 界面，一般而言，普通 PC 允许几个 IDE 界面与装置？
    - IDE 为用来传输硬盘数据的一个汇流界面；
    - 共有 IDE1, IDE2 ，分别有 master 与 slave 所以共四个 IDE 装置支持！

- IDE2 的 master 之第一个 logical 磁盘中，其装置代号（文件名称）为何？
  /dev/hdc5

- 在硬盘分割 (Partition)时，最多有几个 primary + extended ？
  Primary + Extended 共四个，其中 Extended 通常只有一个！(更详细的硬盘与 MBR 可以参考 这里 这篇讨论)

- 若在分割的时候，在 IDE1 的 slave 硬盘中，分割『六个有用』的扇区（具有 filesystem 的），此外，有两个 primary 的扇区！请问六个扇区的代号？

- /dev/hdb1(primary)
- /dev/hdb2(primary)
- /dev/hdb3(extended)
- /dev/hda5(logical 底下皆为 logical)
- /dev/hda6
- /dev/hda7
- /dev/hda8

  请注意，5-8 这四个 logical 相加的总和为 3!

- 一般而言安装 Linux 至少要有哪两个 partition 呢?
  - 根目录 / (root)
  - 虚拟内存 Swap

- 一般而言，在 RAM 为 64MB 或 128 MB 的系统中， swap 要开多大?

  约两倍的 RAM ，亦即为 128 MB 或 256 MB ，可获得较佳效能!

- 光驱与软盘机的代号（装置文件名称）?
  - /dev/cdrom
  - /dev/fd0

- 网络卡（Network Interface Card, NIC）的代号?
  - /dev/eth0

- 预设的 Linux 档案格式为何? 又，目前常用的 Journalling 档案格式为何?
  - Ext2
  - Ext3, Reseifs

- 安装 Linux 的主要流程?
  - BIOS（决定由 cdrom 或 floppy 开机，并加载 PC 硬件信息）
  - 载入 install kernel loader :
  - 收集硬件信息
  - Hard Disk 之 Partition
  - Formation
  - Softpackage 的选择
  - 开始安装!

- 什么是 GMT 时间? 台北时间差几个钟头?

  GMT 时间指的是格林威治时间，为标准的时间，而台北时间较 GMT 快了 8 小时!

- Tap, SCSI 硬盘, RAID, printer 的装置代号?
  - Tap : /dev/ht0 (IDE), /dev/st0 (SCSI);
  - SCSI H.D.: /dev/sd[a-p],
  - RAID : /dev/md[0-15];
  - printer : /dev/lp[0-2]

三、简易 KDE 的使用(X-Window)

- 若以 X-Window 为预设的登入方式，那请问如何进入 Virtual console 呢？
  可以按下 [Ctrl] + [Alt] + [F1] ~ [F6] 进入 Virtual console (共六个)；
  而按下 [Ctrl] + [Alt] + [F8] 可回到 X-Window 的 desktop 中！

- 如何修改进入 Linux 时候的 run-level ？又 run-level 0, 1, 3, 5 各代表什么意思？
  - 修改 /etc/inittab 里头的设定即可；
  - 0：重新开机(如 init 0)；
    1：单人维护模式，没有网络功能；
    3：纯文字接口登入，多人多任务环境；
    5：X-Window 登入模式，多人多任务模式。

- XFree86 在 X-Window 中的角色为何？
  基本角色是控制显示相关硬件的核心角色，也可以说程序 X-Window 的 Server ，此外，KDE 这个 Window management 则是 X-Window 的 Client 呦！因此， XFree86 若死掉了，那么 KDE 就无法被启动！

- XFree86 的主要设定档在哪里？
  就是在 /etc/X11/XF86Config-4 这个档案！

- KDE 预设提供多少个 Virtual desktops ？
  预设是提供四个，就是在进入 KDE 之后，最下方的 bar 上面有 1,2,3,4 那个字样的那个咚咚！

四、在线求助 (Help)

- 在 Linux 系统中，安装的套件 (RPM 安装) 预设的文件数据放在那个目录底下？
  /usr/share/doc

- 查看 Linux 系统的在线求助可以使用什么指令？
  man command
  info command

- 使用 man 来查询在线求助的时候，若要查询类似参数档(如/etc/passwd)需如何？
  man 5 passwd 或是例如 syslog.conf 则： man 5 syslog.conf 那个 5 即是大部分的 config 档案的查询。

- 使用 man 查询 command 的所有相关指令时，需要下什么参数？
  man －k command 例如 man －k passwd

- man page 的 man path 变量，记录在那个档案中呢？！
  /etc/man.conf

五、vi 的使用

- 在 linux 底下最常使用的文书编辑器为 vi ，请问如何进入编辑模式？
  - 在一般模式底下输入： i, I, a, A 为在本行当中输入新字符；（出现 －Insert- ）
  - 在一般模式当中输入： o, O 为在一个新的一行输入新字符；
  - 在一般模式当中输入： r, R 为取代字符！（左下角出现 －Replace-）

- 如何由编辑模式跳回一般模式？
  [Esc]

- 若上下左右键无法使用时，请问如何在一般模式移动光标？
  h, j, k, l

- 若 [pagedown] [pageup] 在一般模式无法使用时，如何往前或往后翻一页？
  [Ctrl] + [f]
  [Ctrl] + [b]

- 如何到本档案的最后一行、第一行；本行的第一个字符、最后一个字符？
  G, 1G, 0, $

- 如何删除一行、n 行；如何删除一个字符？
  dd, ndd, x 或 X （dG 及 d1G 分别表示删除到页首及页尾）

- 如何复制一行、n 行并加以贴上？
  yy, nyy, p 或 P

- 如何搜寻 string 这个字符串？
  - ?string (往前搜寻)
  - /string (往后搜寻)

- 如何取代 word1 成为 word2，而若需要使用者确认机制，又该如何？
  - :1,$s/word1/word2/g 或
  - :1,$s/word1/word2/gc （需要使用者确认）

- 如何读取一个档案 filename 进来目前这个档案？
  :r filename

- 如何另存新档成为 newfilename？
  :w newfilename

- 如何存档、离开、存档后离开、强制存档后离开？
  :w； :q： :wq； :wq!

- 如何设定与取消行号？

  :set nu

  :set nonu

六、档案与目录管理

- 请问底下的目录与主要放置什么数据？
  - /etc/：几乎系统的所有设定档案均在此，尤其 passwd, shadow
  - /etc/rc.d/init.d：系统开机的时候加载服务的 scripts 的摆放地点
  - /boot：开机设定档，也是预设摆放核心 vmlinuz 的地方
  - /usr/bin, /bin：一般执行档摆放的地方
  - /usr/sbin, /sbin：系统管理员常用指令集
  - /dev：摆放所有系统装置档案的目录
  - /var/log：摆放系统登录档案的地方

- 『旧的』 Unix 系统与 Linux 系统的『文件名』最多可以容许几个字符？

  14, 255

- 什么是绝对路径与相对路径，要由 /usr/share/doc 进入到 /usr/share/man 由相对路径与绝对路径的写法各为何？
  - 绝对路径绝对由 / 开始写起，相对路径则非由 / 写起；
  - cd /usr/share/man ; cd ../man

- 在非为根目录的任何一个目录中，下达 ls -al 时，均会有『.』及『..』这两个目录，请问分别代表什么？
  - . ：代表本目录
  - .. ：代表上层目录

- 显示、变换目录的时候，使用什么指令？

  ls, pwd 为显示； 变换目录用 cd

- 新增目录、移除目录移动目录与拷贝目录有什么指令可用？
  - 新增： mkdir 目录,
  - 移除： rmdir 目录(但是该目录内必须要已经清空了), rm -rf 目录,
  - 移动： mv directory1 directory2
  - 拷贝： cp -r directory1 directory2

- 如何查看一个档案的『内容』（不要使用 vi 的情况下）

  cat, tac, more, less, head, tail, nl, od(查看二进制制)

- 什么是 hard link 与 soft link 的档案？有何不同？
  - Hard Links：在做成 hard link 档案时，系统会占用掉一个 inode ，由连结档案可发现其 link 字段多使用了一个 inode ，当源文件被删除的时候，该源文件的内容将继续保留在其它的 Hard Links 档案中；但所有 Link 占用的硬

盘总量仅占一个档案的容量大小！(但 Hard link 不能连结不同 filesystem 的档案)
    - Soft Links：类似快捷方式，当原始档被删除，soft link 档案将找不到原始档了！

- 如何在 root 的家目录下建立一个 /bin 的连结快捷方式

  ln –s /bin /root/bin

- 若有一个连结档为 testing 连结到 test1.sh ，请问 copy testing 到 /tmp 底下，『cp testing /tmp』则 /tmp/testing 这个档案会是：(1) 连结档，连结到 test1.sh ；(2) 一般档案，为内容与 test1.sh 相同？

  答案为 (2)

- 当一个档案属性为 –rwxrwxrwt 则表示这个档案的意义为？

  任何人皆可读取、可写入，但是不可删除该档案(或目录)，除了 root 与档案或目录拥有者有权可以删除。

- 我需要将一个档案的属性改为 –rwxr-xr-- 请问该如何下达指令？

  chmod 754 filename, chmod u=rwx,g=rx,o=r filename

- 在 /usr/bin/passwd 这个档案的属性为 –r-s--x--x 请问代表什么意思(s)

  那个 s 代表为 SUID ，当使用者使用这个档案进行工作的时候，将会具有该档案拥有者的权限！

- 如何改出 –rwxr-sr-x 与 –rwxr-xr-t 这个属性？

  chmod 2755 filename, chmod 1755 filename

- 如何查看一个档案的类型 (type) 例如纯文字文件、执行文件与 setuid 档案等？

  file filename

- 若一个使用者的 umask 为 033 ，请问他建立一个新的目录与档案时的预设属性为？
    - 目录：744 ? -rwx-r--r--
    - 档案：633 ? -rw--wx-wx

- 若我需要更改一个档案的拥有者与群组，该用什么指令？

  chown, chgrp

- 如何将一个档案的修改日期改成目前的时间？

  touch filename

- 如何搜寻一个档案？

  which (仅用于指令搜寻), whereis, locate, fine

- Linux 预设的档案系统为何？此外，常用的 Journalling 档案格式有哪些？

  Ext2, Journalling 有 ext3 及 Reiserfs 等

- less 跟 more 有什么不同？
  less 可以翻页，但是 more 不行！

- 在比较两个档案的异同时，常使用的是 diff 与 cmp ，请教两者有何不同？
  diff 为一行一行比较， cmp 为一个字符(character)一个字符比较

六.1、磁盘挂载与虚拟内存问题

- 我要如何查看 Linux 系统当中所有已经挂载的硬盘容量与 inodes ？
  df －k, df －i,

- 我要如何查看目前所在目录的所有档案占用的硬盘空间，此外，如何仅输出结果？
  du -k, du －i, du -s

- 如果扇区 /dev/hda3 有问题，偏偏他是被挂载上的，请问我要如何修理此一扇区？
  umount /dev/hda3
  fsck /dev/hda3

- 承上题，那么有问题的档案将被移动到那个目录下？
  lost+found

- 试说明新增一个 partition 在 /dev/hdb 当中，且为 hdb5 时，并挂载上 /disk2 ，需要哪些步骤？
  - fdisk /dev/hdb 按 n 新增，按 e 新增 extended ，再按 n 新增 logical
  - mke2fs －b 2048 /dev/hdb5
  - mkdir /disk2
  - mount －t ext2 /dev/hdb5 /disk2

- 如何新增加 64MB 的虚拟内存 swap ？
  - dd if=/dev/zero of=/tmp/swap bs=4k count=16384
  - mkswap /tmp/swap
  - swapon /tmp/swap

- 如果要设定一个新挂载上的扇区，令他可以在开机的时候被挂载上，应该编辑哪一个档案？
  - 先 /etc/fstab
  - 再 mount －a

- 简易说明 quota 的设置流程？
  - 编辑 /etc/fstab， 加入 usrquota and/or grpquota
  - reboot
  - quotacheck －avug （会产生 aquota.user(group)）
  - quotaon －aug
  - edquota －u username
  - edquota －g groupname

- repquota －vu /dev/hd[a-d][1-16]

七、BASH SHELL

- 简单说明 bash shell 的功能特征：
  - 命令记忆功能 ~/.bash_history
  - 命令别名功能 alias
  - shell scripts 功能
  - 命令与文件名称补全功能 <tab>
  - 工作控制功能 jobs

- 在设定变量中，主要的规则为何？
  1. 变量与变量内容以等号来连结；
  2. 等号两边不能直接接空格符；
  3. 变量名称只能是英文字母与数字，但是数字不能是开头字符；
  4. 若有空格符可以使用双引号『 " 』或单引号『 ' 』来将变量内容结合起来，但须要特别留意，双引号内可以保有变量，但是单引号则仅为一般字符；
  5. 必要时需要以跳脱字符『 \ 』来将特殊符号（如 Enter, $, \, 空格符, ' 等）变成一般符号；
  6. 若该变量为扩增变量内容时，则需以双引号及 $变量名称如：『 "$PATH":/home』继续累加内容；
  7. 若该变量需要在其它子程序执行，则需要以 export 来使变量可以动作，如『export PATH』；
  8. 通常大写字符为系统预设变量，自行设定变量可以使用小写字符，方便判断（纯粹依照使用者兴趣与嗜好）；
  9. 取消变量的方法为：『unset 变量名称』

- 显示环境变量与所有变量的指令为何？
  env, set,

- 在环境变量当中，『使用者家目录、主机名称』的变量名称为：
  HOME, HOSTNAME

- 如何让一个变量可以持续到下一个程序 (PID) 去？
  export variable

- 在变量的设定当中，双引号与单引号有何不同？
  双引号里面可以包含变量，单引号谨代表一般字符！

- Command1 \`command2\` 这个指令代表什么意思？
  在这一行当中， command2 会先执行，而输出的结果会给 command1 当作参数。
  cd /lib/modules/\`uname -r\`/kernel 是最常使用的一例

- 在命令重导向当中， > 与 >> 有什么不同？

  > 会将导向的 file 覆盖，

  >> 则是增加！

- 若要将错误的输出导向『不见了』的装置，该如何是好？

  command 2> /dev/null

- 在管线指令中，我要将 last 输出的结果显示 root 的『登入次数』， 如何做？

  last | grep root | cut -d " " -f 1 | wc -l

- 如何设定一个变量名称为 name ，使其显示为『 VBird’s testing variable. 』？
  - name=VBird\’s\ testing\ variable.
  - name=”VBird’s testing variables.”
  - echo $name

- 请问 bash shell 的环境变量与自己的个人变量可以登入便设定的档案？
  - Open Linux：

    /etc/profile, /etc/config.d/shells/bashrc, ~/.profile, ~/.bashrc,
  - Red Hat ：

    /etc/profile, ~/.bash_profile, ~/.bashrc,

- 如何查询曾经操作过的指令？如何执行第 26 个操作过的指令？上一个操作过的指令？
  - history
  - !26
  - !!

- 如何设定命令别名？使 lm 与 ls -al|more 功能相同

  alias lm=’ls -al|more’

- 如何将 /bin 的所有信息输出到 /home/testing.dat 这个档案？

  ls -al /bin 1> /home/testing.dat

- 在上一题中，若还要输出到屏幕上，要如何是好？

  ls -al /bin | tee /home/testing.dat

- 执行 find ，并将正确的结果输出到 right 而错误结果输出到 error 当中？

  find / -name test 1> right 2>error

- 如何在指令列模式中将 /root/.bashrc 的资料加在 /home/col/.bashrc 当中？

  cat /root/.bashrc >> /home/col/.bashrc

- 不要显示执行的结果要如何是好？

  command > /dev/null 2>$1

- 列出这个月曾经登入主机的使用者信息即可，不需要重复？

  last | cut -d” “ -f1 | sort | uniq

- 万用字符当中， *, ?, [] 各代表什么意思?
    - o * 代表 0 到无限多个字符;
    - o ? 代表一个任意字符;
    - o [] 代表一个字符，这个字符在某一个限制范围内。

七.1、SHELL SCRIPTS

- 什么是 shell scripts ? scripts 有何功用?
    - o 简单的说， scripts 就是一个内部含有多个或复杂的 command 的纯文本文件;
    - o scripts 可以进行 program 的功能，但速度上较 C 慢了点!

- 要撰写可以经由键盘输入的 scripts 时，常使用到的指令?

  read variable

- 在 shell script 当中，在控制式中， && 与 || 代表什么意思?

  分别代表『和(同时成立)』及『或(仅其一成立就成立)』的意思!

- 在使用循环的时候，常使用到的控制式为哪三个?

  for, while, until

- 执行 scripts 的方法有哪两种?
    - o 使用 sh script 或
    - o 以 chmod 增加 scripts 的属性为可执行， chmod 777 script 并执行 scripts。

- 如何宣告一个变量成为整数型态?

  declare -i variable

- 为何在 shell script 前面都要宣告 shell 呢?

  宣告 shell 方能让 script 了解该内容需要以何种 shell 来执行!目前 Linux 通常宣告 /bin/bash 这个 shell ，然而若在非 shell 环境中，又没有宣告 shell 类型时，可能会造成 script 无法执行的情况。

- 在判断式中，如何判断一个档名是否存在?

  if [-e filename] 若存在则回传值为真!

- 在执行一个 scripts 时，在 scripts 内的变量， $0, $1 代表什么?
    - o $0 代表 scripts 的档名;
    - o $1 代表第一个 parameter，例如 /etc/rc.d/init.d/xinetd start 那个 start 的变数即为 $1

- 若要写一个既定的 parameter （如 start ），用哪一个判断式较简单?

  可以使用 case …. Esac 的语法较为简单。

- 如何印出目前系统中的所有账号，并加以排序且输出到 /tmp/account 中？
  cut -d ‘:’ -f 1 /etc/passwd | sort > /tmp/account

- 如何在不执行 scripts 的情况中 debug ？
  sh -n scripts

八、基本账号管理

- 如何新增一个使用者 username，且该使用者没有家目录？
  useradd -M username

- Linux 使用者的账号、密码与群组的名称档案放在哪里？
  - /etc/passwd
  - /etc/shadow
  - /etc/group

- 建立新使用者的预设家目录内容在那个目录中？
  /etc/skel

- 建立一个新使用者时，其使用到的相关档案有哪些？
  - /etc/default/useradd,
  - /etc/login.defs,
  - /etc/skel/
  - /etc/passwd,
  - /etc/shadow

- root 的 UID 与 GID 各为何？
  皆为 0

- 如何让一个使用者不能登入主机，但是可以收信？
  将 /etc/passwd 最后一栏代表 shell 的名称改为 /bin/false

- 试说明一个使用者登入系统的流程？
  1. 登入：使用 /bin/login 程序，并输入 ID 与 passwd ；
  2. 确认密码：搜寻 /etc/passwd, /etc/shadow 确认密码！并取得使用者的相关讯息。
  3. 查核 pam 登入模块：这个需要视主机的设定而定！
  4. 取得并执行 shell ：由 /etc/passwd 取得 shell 之后，并执行 shell ，以 bash 为例，将读入： /ete/profile ? ~/.bash_profile (或 .bash_login 或 .profile) ? ~/.bashrc ? 注销的时候执行 ~/.bash_logout

- 在 /etc/shadow 当中的日期设定中，其数字代表的意义为何？
  由 1970 年开始计算，故 1970 年 一月一日为 1 ，2002 年 1 月 1 日为 11689

- 如何变更使用者的一些属性？

  usermod, chfn, chsh, passwd

- 如何视察一个使用者 username 所拥有的群组呢？

  groups username

- 试说明如何手动增加一个使用者 username 与群组 groupname？
  1. 先看看 /etc/skel 当中，预设要给 user 的家目录内容；
  2. 新增 group ： groupadd groupname
  3. 新增 user ：useradd -m -g groupname username
  4. 给予密码： passwd username

- 试说明 /etc/passwd 这个档案的内容与格式：

  账号；密码；UID；GID；说明的内容；家目录；SHELL

- 使用 id 这个指令时，可以显示什么讯息数据？

  UID, GID 跟 GROUP

- 可以控制使用者使用主机资源的预设档案在哪里？

  /etc/security/limits.conf

- 基本的压缩指令有哪些？

  tar, gzip, compress

九、开机程序

- 试说明开机流程：
  1. BIOS
  2. MBR (boot loader)
  3. Loader (lilo or grub, stage 1 and stage 2)
  4. Kernel loader (/boot/vmlinuz)
  5. init process (读取 /etc/inittab 取得 run-level)
  6. 开始执行 /etc/rc.d 内的 scripts
  7. 执行 /etc/modules.conf 内部的额外 kernel 模块
  8. 执行 /etc/rc.d/rc[1-6].d 的这些 run-level 内的 scripts ！
  9. 执行 /bin/login 等待使用者登入!

- 开机时后的加载讯息可以看哪里？

  dmesg

  grep “kernel” /var/log/messages

- 改变 run-level 或关机的指令

  shutdown,

  halt,

reboot,

init or telinit

- 改变登入 Linux 时候预设的 run-level 要改那个档案？

  /etc/inittab

十、程序与资源管理

- 如何查看目前的程序？

  ps， top,

- 如何查看目前的内存使用状况？

  free， top,

- 目前的工作如何丢到背景中？

  [Ctrl] + z

- 如何取得目前背景中的工作，且将他拉回前景中？

  jobs　及 fg %number

- 如何设定一个程序在开始执行时候的优先值？

  nice －n [number] [command]

- 什么指令可以修改一个正在执行的程序之 nice 值？

  top， renice

- 在例行性命令中，使用的两支 demane 是什么？

  atd,

  crond

- 使用 crontab 这个『指令』的时候，如何可以查看目前的工作与删除目前的工作？

  编辑： crontab －e,

  查看： crontab －l,

  删除： crontab －r

- 常用的 kill 指令之 signal 当中， 1, 9, 15 代表什么？
  - 1　重新读取一次 config file
  - 9　强制删除此一执行程序
  - 15 terminal 结束的意思！

序言

最近更新日期：2003/09/18

关于本书：

笔者在 2002 年底写完了『鸟哥的 Linux 私房菜 -- 基础学习篇』以来， 接到很多朋友们的鼓励，很感谢大家的支持呐！另外也有很多的来信要鸟哥将接下来的服务器架设篇赶紧给他写一写， 其实鸟哥也觉得应该早点将架站篇的内容也给他完成的，这样才能够连贯的起来啊！不过， 架站篇比起基础篇来说，说实在的，要复杂的很多！怎么说呢？如果说 『架站篇』就单纯的讲架站流程，呵呵！ 那么随便几个钟头就可以将好几个不同的服务器的架设方法写完了，因为这些架设的方法都有资料可以参考，而且， Linux 本身就有极为丰富的说明文件了！所以说，如果架站篇只是写架设的流程，鸟哥实在觉得不够过瘾～

而且，网站的架设其实由网站成立之初的规划开始，到实际硬盘的分割 (partition)，软件的选择与安装，架设完成之后的后续监测与维护，还有那个挺重要的备份工作等等， 其实是需要『一贯作业程序』的，什么是 『一贯』作业程序呢？其实就是上面所有的工作都需要 『全部一起搭配来思考』的意思，您不能单纯的只想到某个比较重要的地方而已。

举个例子来说好了，假如您想要架设一个给学生使用的档案服务器 (File Server, 一般使用 SAMBA)，那么在架设之前，您想到，嘿嘿！学生数很多，所以我的硬盘要很大，也因此，您就在 Linux 上面安装了一款 120 GB 的硬盘，然后，很高兴的将硬盘分割为『只有 / + Swap 』这样的懒人分割法。等到实际上机运作之后，却发现有的学生占用了主机硬盘好几 GB 的空间，使得其它同学无法使用主机所提供的档案服务！等到发现这样的情况，要再加以使用 Quota 解决的时候，却发现当初硬盘规划的不好(因为只有 / 啊！)，使得无法进行较佳的 quota 设定。还有，备份也成了一个大问题，因为没有多余的额外空间来存放备份数据了～ 这样可以了解一贯作业程序了吧？！是的！您必需要视自己的需要来规划主机，并且规划的时候， 就需要从头到尾的做一个整体的设计了呢！

这些整体搭配的架设网站流程，其实都是网站架设者所需要进行的工作，不过， 目前大家常见的书籍在这方面谈的都不多，而大多纯粹的讲一些技巧性的架站手法，嗯！ 这样的书籍也是有需要的啦！不过，鸟哥认为，如果可以谈的更深入一点， 将网站从架设之前所需要考虑的事情以及架设完成后的后续工作都一起谈进去的话， 那么对于网站维护者来说，应该会有一个比较容易入门的管理与维护方法的认识吧！ 所以啰，这一本书的内容，除了基本的网站架设流程之外， 还会谈到许多的网络基础概念、网站的相关维护技巧以及鸟哥平时维护主机时所认为较佳的维护经验谈。

谁适合这本书：

这本书既然是谈论比较深入的架站规划、流程、技巧与维护等工作，那么比较基础的 Linux 操作与相关的 Shell 语法，在这本书里面就不可能谈论的很多，毕竟， Linux 基础篇 已经完成了，没有必要在这本书里面再次的重复提及的。 所以，当您尝试阅读这本书的时候，请注意，您最好已经具备有 Linux 操作系统的相关知识，以及文字接口 (BASH Shell) 的相关技巧，还有，必需要能够了解一些 Unix-Like 的工作流程，例如登录文件的产生与放置的地点、服务的启动与关闭方式、 工作排程的使用方法、以及其它种种相关的事项。也就是说，如果您从未接触过 Linux ，那么建议您由『鸟哥的 Linux 私房菜 -- 基础学习篇』开始 Linux 的探索历程，否则，这本书对您而言，可能会过于难以理解。

另外，这本书的内容很多时候会提到一些简单的概念而不是僵化的流程， 尤其每个人对于网站的要求都不

相同，也就是说，每个人的网站其实都是带有个人风格的， 因此僵化的流程并没有太大的意义～只要能够依据这些简单的概念来进行网站的架设，鸟哥认为， 您的主机设定应该都不会有太大的问题。怕的是什么呢？都没有碰过 Linux ，却想直接参考架站的程序来完成网站的架设的朋友，这些朋友最容易忽略后续的维护与管理了！ 这也容易造成网站的不稳定或者是造成被网络怪客 (Cracker) 入侵的问题啊！

这本书主要的目的是引导使用者进入 Linux 强大的网络功能的世界， 书内的范例都是鸟哥自己实际测试过没有问题才写上来的，不过，毕竟每个人的网络环境与操作习惯都不相同， 因此，鸟哥不敢说我书内的范例一定可以在您的系统上操作成功的！ 然而，书内都会提到一些基本概念的问题，只要理解这些基本的概念，并且对于 Linux 的操作熟悉，相信您一定可以利用书内的范例来开发出适合您自己的服务器设定的！不过，对于没有碰过 Linux 的朋友，还是建议从头学起，至于为什么一定得从头学起，在本书的第一章内会仔细谈论喔。

章节安排：

本书在章节的规划上面，主要分为三大部分，分别是『网络基础篇』、 『简易防火措施篇』与『服务器架设篇』，前两篇的所有内容与『服务器架设篇』都具有很高的相关性，例如 NAT 服务器就与『简易防火措施篇』内的简易防火墙有高度相关！ 所以，您在开始服务器的架设之前，请务必将前面两篇共十二章先念过一遍才好呐！

在『网络基础篇』当中，我们会介绍简易的网络基础， 这包含了硬件的选择与布线。此外，还有在 Linux 上面连上 Internet 的方法，以及在 Linux 发生无法连接因特网的问题时，简易的查验方法。 看完了这一篇之后，您的 Linux 不论以何种方式来进行 Internet 的连接，就应该都不成问题啰，而且，鸟哥希望看完这一篇之后，您可以了解 Linux 的网络问题，并自行解决喔！

在『简易防火措施篇』中，我们会简单的介绍 Linux 的强大网络功能下，可能会发生的网络入侵问题。 接下来，了解了问题后，当然就是需要来解决他啰！所以，我们会就 TCP/IP，port， 套件漏洞的修补与防火墙等来推敲一下，该如何做好 Linux 主机的防备呢？！ 『没有永远安全的主机』是正确的言论，所以， 即使您的主机只是一个小小的网站，也千万不能忽略这个防火墙的认识喔！

在『服务器架设篇』当中，我们会介绍 WWW， dns， mail， ftp， dhcp， samba.....等等的服务器，在这一篇内的文章您就可以跳着看了！ 因为不想玩的服务器，当然就不需要去看他啊！唯一一个例外的就是 DNS 服务器 (领域名称服务器) 啰， DNS 是所有服务器能否正常工作的基础，因此，您虽然不需要架设 DNS 服务器，但是得了解 DNS 的整体运作流程呢！

章节的安排主要仍然是由浅入深来进行编排的，因此，还是希望读者们可以由前面慢慢的往下看， 不要着急的直接翻到后面去抄一些架设流程喔！而且，几乎每一章节后面都会具有一些简单的课后练习题， 这些练习题有的是鸟哥参加过的考试内容，有的是鸟哥想到的一些数据，很适合大家思考喔！ 不要错过这些练习题的训练喔！

服务器简介：

在这本书里面提到相当多的服务器架设，这里先就几个常见的服务器介绍一番， 提供大家先有个认知，后续请读者们自行到该章节阅读更进一步的数据喔！

- Router (路由器)：
  我们在设定网络时都会订定通讯闸(Gateway)，这个 Router 玩意儿就是 Gateway 咯。Router 可以用来沟通两个不同的网段，使得资料可以互传， 是网络上相当重要的一个设备；

- Firewall (防火墙)：
  Linux 上的防火墙主要是由 iptables 这个核心功能(或者可以称为机制)所设定达成的，利用每个数据封包要进入 Linux 系统之前的分析与过滤，来剔除危险的信息，保护我们 Linux 主机的安全；

- Telnet & SSH (远程联机服务器)：
  我们可以在个人计算机以 telnet 这个程序以及 putty 这个软件来连接到主机，只要连接到主机后，整个屏幕上的作业就像您坐在主机前面工作一般的方便！

- NAT (Network Address Translation)：
  简单的想，NAT 服务器就是『IP 分享器』，善用 NAT 还可以达成将服务器架设在 Intranet 的功能！

- NFS (Network FileSystem)：
  在 Linux 与 Linux 上面分享档案最方便的工具就是 NFS 了！他可以将远程的 Linux 主机所分享出来的『目录』挂载到自己的系统下，作业起来就像自己 Linux 系统的一块 Partition 一般的好用！

- DHCP (Dynamic Host Configuration Protocol)：
  如果您有管理超过 10 部以上的个人计算机在您的局域网络内，那么使用一部 DHCP 主机来统一分配区域内所有个人计算机的 IP 以及相关的网络参数，是一个很不错的解决方案！

- DNS (Domain Name System)：
  DNS 的概念相当的重要，您可以透过 Internet 上面的任何一部 DNS 主机来达成主机名称与 IP 的对应，此外，DNS 与邮件主机的相关性也很高喔！总之，如果您管理多部主机， 且这几部主机的主机名称需要自行掌控，那么就需要架设 DNS 服务器了；

- WWW (Web Server)：
  在 Linux 上面使用 Apache 这个套件来达成 WWW 的服务器架设，如果同时以 PHP 及 MySQL 来设定您的 WWW 服务器，呵呵！您的网页会很热闹啊！

- SAMBA (档案服务器)：
  在 Linux 与 Linux 的档案分享使用 NFS ，那么在 Linux 与 Windows 的档案分享则是以 SAMBA 咯！架设 SAMBA 之后，您可以透过 Windows 系统的『网络上的芳邻』来连上 Linux 主机分享资源喔！

- Sendmail & Postfix (邮件主机)：
  想要收发 e-mail 吗？！那么就得要架设邮件服务器了！目前几乎所有的邮件服务器原理都以 sendmail 为依据，所以不论您使用哪一套邮件服务器软件，建议 sendmail 的运作还是需要熟读的！

- Wu-FTP & Proftpd & vsftpd (档案传输服务器)：
  档案传输服务器的架设软件真的很多，不过传统上， Linux 是以 Wu ftp 为大宗，然而因为安全

性与功能性，建议使用 proftpd 与 vsftpd 呢！我们可以透过 Client 端的 cuteftp 等图形化接口连接上 Linux 的 FTP 功能来上传/下载档案呐！

- Proxy (代理服务器)：
  利用 Proxy 可以达到简易的控制您局域网络内可前往浏览的网站，如果加上分析软件，更可以控管您局域网络内部的 WWW 浏览行为！此外，如果局域网络内的计算机数量多，那么 Proxy 还可以达到节省频宽的功能呐！

- NIS (Network Information Services)：
  NIS 可以统一管理不同主机的账号，让不同的主机可以具有相同的账号与密码，如果搭配 NFS 的话，功能就更多样化了！

- NTP (Network Time Protocol)：
  平常您如何调整自己的手表时间？看电视、听广播对吧！那如果是您的网络主机想要校时呢？那就利用 NTP 主机的功能吧！

- APT (Advanced Package Tool)：
  套件升级是很重要的！如果能够架设一部 APT 的话，那么所管理的所有 Linux 主机的升级就会相当的便利喔！但如果所管理的 Linux 并不多，这个 APT 就不需要架设了！

书上的内容至少有上面那么多的咚咚可以查阅，祝大家学习愉快！

感谢：
这本书的内容并不是三天两夜就完成的，书内最早的数据可以追溯 2001 年，所以说穿了就是鸟哥在 Linux 服务器方面的一些成长经验。 在最早的文章发表过程当中，受到相当多朋友的检验，并且让鸟哥得以得到相当多很棒的建议， 而据以修改网站上的文章！这都要感谢 Sutdy Area 与 TnLUG 的朋友群：Netman, 梁枫, 蔡大哥, duncan, 逸晨, Jerry Wu, Wilson, damon 等，以及酷学园台北帮的伙伴们：zman, ericshei, 日京三子等，还有很多 Linux 的前辈们：小州先生与果正兄等，当然还有更多不及备载的朋友们的协助，感谢您们的扶持！此外，当然还有更多的读者们的支持，没有大家的支持，就没有鸟哥的私房菜啊！

事实上，这本书从签约到成册的短短两三个月的过程中，在鸟哥的身上发生了很多突发的事件， 这包括学业无法顺利完成的遗憾以及母亲车祸离开人世的伤痛。还好身旁好友玉南、士杰、景阳以及阿毛的鼓励， 小弟俊明、秀明、瑞明的相互扶持，以及邱爸爸、邱妈妈一家人温暖的照应， 当然还有女友慧真在生活方面的照料，使得任性的我可以暂时离开伤痛，继续往 Linux 的世界前进。最后，仅将这本书献给我在天堂的妈妈，妈妈，我会继续努力的。

鸟哥 2003/09/18

架站之前所需的技能分析

最近更新日期：2006/06/06

很多刚接触 Linux 的朋友常常会问的一句话就是：『我学 Linux 就是为了架站，既然只是为了架站，为什么我还要学习 Linux 的其它功能？例如：例行性工作排程、Bash Shell 、干嘛去认识所有的登录档等等，我又用不到！此外，既然有好用的 Web 接口的 Server 架设软件，可以简单的将网站架设起来，为什么我还要去学习 vi 手动的去编辑一些设定档？干嘛还需要去理解他的原理？』 上面这些话对于刚刚学会架设网站的人来说，真是替他们道出了一个新手的心声啊！ 不过，对于任何一个曾经有过架设公开网站的朋友来说，上面这些话，真的是会害死人～

要知道，『架站容易维护难』啊！更深一层来说，『维护还好、除错更难啊！』 架设一个网站有什么难的？即使您完全没有摸过 Linux ，只要参考鸟哥的书籍或者是网站，而且一步一步照着做， 包准您一个下午就可以架设完成五个以上的网站了！所以说，架站有什么难的？ 但是，要晓得的是，这样的一个网站，多则三天，少则数小时，立刻就会被入侵了！ 此外，被入侵之后，或许可以藉由一些工具来帮您将 root 的密码救回来，可惜的是，这样的一个网站还是有被做为中继站的危险存在的！此外，如果您使用工具 (例如 Webmin) 却怎么也架设不起来某个网站时，要怎么解决？如果您不懂该 Server 的运作原理与 Linux 系统的除错讯息，那么难道只能无语问苍天？不要怀疑这种情况的可能性，参考一下 BBS 上面的留言就可以很清楚的知道这种情况的存在有越来越明显的趋势呢！

所以说，架站之前还是有一些基本的技能需要学会的！而且这些技能是『一旦学会之后，真正是终身受用啊！』 只要花一个学期(三~六个月)就能学会一辈子可以使用的技能，这个学习的投资报酬率真是太高了！所以，一开始的学习不要觉得苦，那真的是值得的喔！^_^

前言

如果有人问你：『Linux 最强大的功能是什么』？ 你大概大家都会回答『是网络功能啊！』， 接下来，如果对方再问：『所以学 Linux 就是为了架站啰？』 呵呵！这个问题可就见仁见智啰！说穿了， Linux 其实就是一套非常稳定的操作系统，那么任何工作只要能在 Linux 这个操作系统上面跑，那他就是 Linux 可

以达成的功能之一啰！所以 Linux 的作用实在不止于网络服务器的架设哟。

举例来说，在 Linux 上面开发跨平台的程序 (program) 诸如大型的数值模式，由于 Linux 的稳定与强大的资源分配功能，使得在 Linux 上面开发出来的程序运作的又快又稳定。此外，诸如 KDE, GNOME 等漂亮的图形接口，搭配诸如 Open Office 等办公室软件，Linux 立刻摇身一变而成为优秀的的办公室桌面计算机了 (Desktop)。所以说，千万不要小看了 Linux 的多样功能呐。

不过，不管怎么说， Linux 的强大网络功能确实是造成 Linux 能够在服务器领域内占有一席之地的重要项目。 既然如此，我们就好好的来探索一下 Linux 的网络世界吧！首先， Linux 到底可以达成哪些网络功能呢？这可就多着咯！不论是 WWW, Mail, FTP, DNS, 或者是 DHCP, NAT 与 Router 等等，Linux 系统都可以达到，而且，只要一部 Linux 就能够达到上面所有的功能了！当然，那是在不考虑网络安全与效能的情况下，您可以使用一部 Linux 主机来达成所有的网络功能。

『哇！Linux 有那么多的功能啊！那么我可以轻轻松松的就架设好一部以 Linux 为操作系统的服务器吗？！』呵呵！很可惜，答案是『否』！ 您无法轻松的就完成一部『堪称完美』的服务器架设，而是必需要很用心， 并且学习很多相关的概念与操作方法后，才能够架设好一部完美的服务器。什么？！要很用心啊！使用 Windows 随随便便就可以架设好好几个服务器了，那么我干嘛要花时间去学习 Linux 来架设服务器呢？

唉！这真是伤脑筋呐！『谁说使用 Windows 架设服务器就很简单』的？！ 就鸟哥的感觉来说，架设 Windows 服务器一点也不轻松。为什么呢？

1. 首先，在尚未进入服务器设定之前，您必需就『授权模式』进行深入的研究，因为， 不同的使用者数量将会影响到您的服务器的『价格！』，光是这一点，就可能让我一个头两三个大了～ 因为，玩工程的，对于『价格』这东西，总是缺乏一点概念啊～好了，经过了授权模式的洗礼之后，
2. 再来进入到服务器的设定方面，呵呵！这个部分可就容易的多了吧？！没错， 确实是按几下鼠标按钮就可以架设好一两个服务器了。不过，『万一』该服务器架设完成后， 并不符合您当初的要求时该怎么办？不会发生这样的事情吗？！当然会发生啦！因为 Windows 为了达成所谓的『亲和性与便利性』，所以在您的设定过程中，他会『很亲和的自动帮您加入某些参数』， 不过可惜的是，这些参数并不见得适合每个人，所以有时候您必需要自行修改这些设定值。偏偏 Windows 服务器大部分的设定档都是一些特殊格式，您无法使用简易的文书编辑器去修改～
3. 再者，更遗憾的是，如果服务器设定出了点小问题，总是无法达成您的要求，设定也都看过了， 那么要如何除错呢？！如果您没有网络的基本概念，以及 Windows 相关的登录档案管理技巧，呵呵！即使 Windows 在设定上做了很多的简化，我想， 您依旧无法设定出适合您自己的服务器的啦！

所以说，不管是 Windows 还是 Linux ，其实，要架设好一部堪称完美的服务器，『基本功课』还是得做的，这包括了：

1. 该操作系统的简易操作，以及登录分析、账号管理、文书编辑器的使用等等的技巧；
2. 网络的基本概念；
3. 防火墙方面的相关知识等等。

而且，每一个项目里面所需要学习的技巧可多着呢！『什么？要学的东西那么多啊？！』是啊！ 所以，不要以为信息管理人员整天闲闲没事干的呐，大家可是天天在出卖知识的，同时， 还得天天应付随时可能会发生的各种漏洞与网络攻击手法呢！真不是人干的工作～～

这么说的话，架站真的是挺难的喔！事实上，架站其实蛮简单的哩！咦！～怎么又说架站简单了？ 不是说架站难吗？呵呵！其实『架站很难』是由于朋友们学习的角度有点偏差的原因啦！ 还记得当初进入理工学院的时候，天天在念的东西是基础物理、基础化学、工程数学与流体力学等基础科目， 这些科目花了我们一至两学期的时间，而且内容还很难呐～都是一大堆的理论背不完。 怪了？我们进理工学院是为了求取更高深的知识，那么这些基础知识学了有什么用呐？！ 呵呵！更高深的知识都是建构在这些基本科目的理论上面的，所以， 万一您基础的科目没有读好，那么专业科目里面提到的基本理论怎么可能听的懂？！

这样说应该就不难了解了吧？！没错！认识操作系统与该操作系统的基本操作，还有那个重要的网络基础，就是我们在架站前的『基础科目』啦！所以说，在进入 Linux 的服务器世界之前，真的不能够略过网络基础的相关知识，同时， Linux 系统的基本技能也必需要能够理解呐！

好了，或许您还是对于 Linux 系统里面『什么是很重要的知识』不甚了解， 果真如此的话，那么我们就举个简单的例子来说明一下啰！底下列出一般的架站流程， 我们由架设服务器的流程当中，来看一看什么是重要的 Linux 相关技能吧！ ^_^。(注：在这一章当中，鸟哥不再就 linux 基础指令进行解析，因为在『鸟哥的 Linux 私房菜 -- 基础学习篇』里面已经详细的介绍过了！ 如果持续的介绍指令，简直是浪费篇幅～所以底下仅介绍一个 Linux 基础学习重要性的分析喔！)

基本架站流程：

虽然不同的 Server 提供的服务并不相同，而且，每种服务的原理也不见得都一样，不过， 每种服务器由规划、架设到后续的安全维护，事实上，整个流程是大同小异的。 所以，底下我们就整个服务器的简易架设流程当中，来分析一下， 为什么了解操作系统的基础对于网站维护是相当重要的呢？

首先，先来分析一下，如果你要架设一个网站时，架设的基础流程是怎样的一回事。大致的流程有点像底下这样：

1. 了解网络基础 ：
   既然要架设网站，如果对于网站最最基本的网络基础知识无法具备的话， 那么.....当然无法管理好网站啦！举个例子来说，不论何种操作系统， 常常会使用到所谓的『网域』的概念，当您发现一个设定为 192.168.1.0/255.255.255.0 时，晓得那是什么鬼东西吗？！ 如果不知道的话，呵呵！绝对无法设定好网站的啦！
2. 了解架站的目的：
   想要架站，架什么站！？架这个站后，要不要对 Internet 开放？要不要提供什么服务给其它外面（指 Internet 上的用户 ）的使用者？提供这些服务时，需不需要做限制（ 例如限制使用者可以使用的硬盘空间或网页、邮件的最大容量 ）？如果要做限制时，需要怎么样选购您的主机硬件？要进行这些规划时，都需要知道架站的目的呢。 不过，如果架站只是为了『练功』而已，呵呵！那就不需要考虑太多了～
3. Linux 安装硬盘规划
   好了，不论您的网站规模有多大，只要是对 Internet 开放的网站，几乎一定都需要硬盘的啊！因为网站的资料需要有地方可以储存呐！ 那么您要如何选购硬盘，还有，硬盘应该进行怎样的分割（ Partition ）比较好？举个例子来说，如果您想要架设邮件主机，那么硬盘应该如何规划呢？给您猜～

4. 了解欲架设的网站服务原理

   这个与刚刚第一点有点类似！也是属于基本原理方面啦！举个例子来说，当您晓得了 Mail Server 的运作原理，自然就比较容易架设成功，此外，也比较容易进行除错！

5. 服务的套件安装、漏洞修补、套件升级...

   好不容易决定了硬盘的规划，并且 partition 与 Linux 这个操作系统都安装好了，接下来自然就是安装我们所想要架设的服务器软件啦！例如：如果要架设 Mail Server ，那么 Sendmail 或者是 Postfix 这两个邮件服务器软件就派上用场啦！咦！ 有两种邮件服务器软件啊？呵呵！当然不止～邮件软件可多的很呢！那我要选择哪一个邮件服务器软件？ 需要考虑安全性、架设的便利性、以及执行的效能与稳定性等等！呵呵！累了吧！

6. 主机设定、启动、观察与除错

   在完成了 Linux 安装，并且将服务器软件安装好了之后，再来当然就是设定啰！ 这个部分就是我们常常看到的一些文件说的比较多的部分啦！在主机的设定当中，其实设定项目并不难， 了不起都是照着文件设定就可以了！不过，由于每个人的主机环境不一样(例如安装的套件版本啦, Linux distribution 的不同啦, 编译器的不同啦，这些都是主机环境的一环！)，所以，同样的一份设定在不同的机器上，嘿嘿，可能不会一定百分之百能执行的呐！ 这个时候，观察主机的登录文件与相关的讯息，并加以进行错误克服 (debug) 的动作可就相当的重要了呢！这也是大家常常会忽略的部分。

7. 客户端设定、观察与除错

   这部份也是需要的，因为有的服务器需要客户端也进行设定才行！例如邮件主机加上身份认证功能时， 就需要在客户端上面设定好身份认证的确认啰！

8. 安全性设定

   网络安全是很重要的，问题是，要怎么样达成网络安全的相关设定呢？ 有哪些档案、套件、指令与数据可以查寻？

9. 服务日志、登录文件与备份管理

   呵呵！毕竟没有人敢说『我的网络是绝对安全的， 我的硬件是绝对没有问题的！』既然如此的话，备份就成了重要的课题了！ 问题是，如何备份呢？使用什么指令来备份？使用什么媒体来备份？需不需要手动来备份？ 还是交给系统自行每日、每周自动备份？！这都是挺重要的！

整个服务器的架设流程大概就如同上面所提的几个步骤啰！那么每个步骤底下与 Linux 系统基本操作有关的数据有哪些呢？我们就分项目来谈一谈吧！

了解网络基础

既然是要架设网站，那么网站最基本的知识，就是那个 OSI 七层协定，至少就需要了解一下啰！虽然只是一个 OSI 七层协议，但这里面包含了：

- 硬件与设备的等级与好坏：例如网络线有哪些等级？CAT 5 是目前常见的 RJ-45 接头的网络线等级，为什么要选择这种网络线？啊什么是 RJ-45 的接头！？如果这个都不知道的话，那么要如何自己将所有的计算机接起来呢？ 呵呵！很难吧！所以，这种基本的硬件配备的选择需要有一定程度的认知喔；
- TCP/IP 与封包的概念：例如 IP, Netmask, Broadcast 的概念与书写的格式等；
- 路由(route)：您的数据封包是如何传送到 Server 上的呢？资料在 Internet 上面是怎么跑的呢？！

- 领域名称系统 (Domain Name System, DNS)： 常常会听到『主机名称』吧！是否我自己设定一个主机名称，大家就可以通过 Internet 连接到我的主机了？当然不是！还需要透过上层 DNS 主机的授权喔！那么如何授权？
- 各种网络设备的选择与搭配：例如什么是 Hub/Switch ？该如何选择？以及什么是下行/上行 512/64Kbits ？什么是并行线与跳线等等;

还有我们最关心的，网络频宽到底是由哪一个设备所限制，以及在局域网络内的布线该如何配置。 哇！说起来好像很麻烦！其实只要稍微深入了解一下基本的原理，就可以约略的厘清您一些网络概念了。 这对于您未来架设防火墙以及维护网站，尤其是发生问题后的问题认定与除错，都有很大的关系呐！ 事实上，不论您是使用什么操作系统来架设您的网站，这个网络基础的部分都是一定要了解的！ 通常不了解网络基础的朋友，在架设网站的时候，最容易发生一些基础设定的错误了！ 所以，这部份的学习千万不能省略啊！

对于网络基础的问题当中，一个最常见的经典案例是： 『192.168.1.0/24』是什么东西？呵呵！这东西代表的是一个『网域』， 这个网域起始到结束的 IP 范围是多少您看得出来吗？！这种表示方法在各种服务器的架设中很常见到， 如果您不了解的话，呵呵！当然无法进行服务器的架设啰 (注：各种操作系统均会出现这样的设定值喔)。还有，如果您的主机明明就可以使用 ping 这个指令去接触远方的主机 (ping IP)，但是就是无法使用 ping hostname 去接触远方的主机，请问，这个原因是什么呢？了解网络基础的朋友一看就知道几乎是 DNS 出问题了，不晓得的朋友就是想破头也得不到答案～

其实这些概念说难不难，但是至少一定得需要学习过呐！ 而且，这些知识在您学会之后，未来再接触到其它不同的操作系统时，呵呵！可让您学习的过程『顺畅无比』呐！ 因为那是最最基本的理论，各个操作系统都是建构在这些网络基础上面，所以，当您换了另外一套操作系统后， 基本上，只有指令的下达方式不一样而已，基本原理是一样的呐！所以说，学习的过程当然会顺畅到底啰！ ^_^。

而且，网络基础会影响到您的网络设定是否正确，这真的很重要呐，因为，如果您的网络不通， 那么即使服务器架设成功了，别人可以看的到吗？！所以说，要架站，真的得对网络基础的部分下一些功夫才行的。关于网络基础这部份的介绍我们会在下一章网络基础时再详加说明喔！

了解架站的目的

架站的目的与您的主机规划是一体两面的！因为我们必需要了解主机未来的用途与使用的客户端数目， 才能够开始选择硬件与分配网络频宽，并且规划我们的 Linux 系统啊！所以说，了解架站目的是很重要的！举几个例子来说好了：

- 如果我们的主机是学校单位提供同学们邮件与 WWW 个人网页的功能，而全校有 5000 多位同学，那么硬件应该如何选择？还有，如果需要对每个同学进行网页空间与邮件容量的限制时， 硬盘又该如何分割？

  在这个例子当中，您必需要了解邮件与 WWW 个人网页的数据所放置的目录在哪里？而由于主机有进行硬盘的容量限制，因此， quota 的原理与设定的技巧就不能不知道啦！此外，既然是架站嘛！申请一个『合法』的主机名称就显的相当的重要，另外， DNS 里面的 MX 标志对于邮件主机的用途也需要特别去了解呢！
- 如果我们的主机仅提供研究室不到十个人的服务，且服务仅开放档案服务器 (File Server)，那么硬件又该如何选择，此外，是否需要注册一个合法的主机名称呢？

因为仅开放档案服务器，那么这部主机应该是在内部了，所以，当然就不是很需要一个合法的主机名称了！ 不过，因为仍然有对局域网络提供档案服务器的功能，因此，硬盘的容量也不能太小，此外， 为避免未来使用者的抱怨，也需要预留空间来进行 quota 与增加硬盘等工作呢！

- 如果我们的主机主要的目的是进行数值模式的运算 (就是有点像是超级计算机在算军事或者天文物理现象的模式)， 那么应该要考虑的是否反而就是在系统安全性与数据传输的速率上面？

  当然啦！这样的一个系统最主要的目的就在于『稳定性』与『速度』上面了！因此，您就需要了解 核心 (Kernel) 的编译技巧， 此外，由于不同的套件在设计最佳化的时候，通常都使用较为常见的 Linux distribution ，因此，对于 Linux distribution 的选择可又重要的多了呢！

由上面的例子来看，不同功能的主机在硬件配备与软件规划上面是不一样的！所以，在进行 Linux 主机的规划之前，建议一定要了解这个主机的功能，甚至最好还能预设一些未来可能加入的功能规划呢！ 那我怎么知道我的主机要有什么样的功能？又什么样的功能可以满足我的需求呢？这当然是需要经验的！ 呵呵！这又得要回到刚刚前一个主题啰！至少需要了解到网络的基础， 这样就能够知道您应该架设什么样的网站啦！

举个例子来说，如果您只是想要让 Linux 达成频宽分享而已，那么 Router 或者是 NAT 的设定即可满足你的需求， 万一您还需要了解每个使用者经常使用的网站，那么最好就是搭配 Proxy 以及分析软件来分析。然而 NAT 不需要很好的硬件配备即可达成，Proxy 如果要效能好一点，就必需要比较高档的硬件来搭配～咦！我怎么知道的？ 这当然是鸟哥已经较为熟悉每种服务所使用的硬件状态与他的服务原理啊。 所以啰！在主机的规划之前，务必要先了解主机的架设目的喔！

那么这部份与 Linux 基础有何关系？！关系大了！

1. 必需熟悉 Linux 的档案配置与架构：

   例如 /boot 放置的是开机核心数据、 /usr 放置的是预设的系统程序以及原始文件与一些 man page 、/home 是一般身份用户的预设家目录所在、 /var 则是登录档、暂存盘、 邮件信箱档案等放置的目录、/tmp 是任何人皆可使用的暂存目录等等，您必需要熟悉 Linux 各个目录的用途，这样在未来进行硬盘的规划时，才能够符合需求！

   举例来说，因为磁盘配额 (Quota) 仅能针对整个 partition 来动作，而如果您想让每个使用者有两种 quota 数值 (例如每人拥有邮件 20MB 以及 WWW 10MB 的硬盘空间) ，就必需要预设针对 /home 与 /var/spool/mail 规划出两个独立的 partition 才行啊！当然啦，使用连结档 (link file) 也是一个可行的方案，不过， 您就得要预留一些硬盘空间来预备使用呐！
2. 硬盘的代号：

   硬盘的代号真的是挺重要的，因为在 Linux 底下，每一个装置都以档案来表示的！ 因此，您必需要了解到硬盘与排线的接法对于硬盘在 Linux 系统中的代号的关系。例如：

	Master	Slave
IDE 1	/dev/hda	/dev/hdb

IDE 2	/dev/hdc	/dev/hdd

3.

上面的磁盘代号只与 IDE 接口的装置有关，什么是 IDE 接口啊！就是目前所谓的 Ultra DMA 66/100/133 的硬盘啊，或者是光驱与刻录机，这些装置均可直接以主机板上面的 IDE 排线接在一起说。还有，目前主流的硬盘已经变成 SATA 接口了，这种 SATA 接口可以仿真成为 IDE 接口，因此，您的 IDE 装置还可能具有 /dev/hde, /dev/hdf 等等呢！ 若是使用 SATA 格式的话，那么该硬盘的代号就可能变成 /dev/sda 之类的档名。 也就是说，使用 SATA 接口的硬盘时，该硬盘代号的名称与 BIOS 内选择的 SATA 格式是有关系的喔！

不过，万一您使用的不是 IDE 装置呢？例如 SCSI 接口的硬盘！那么就得由 /dev/sda 开始算起了！同时，目前很常见的 USB 接口的随身碟或者是 USB 的外接式硬盘，他的代号也是 /dev/sda 开始算起喔！除了这些基本概念外，在硬盘的 partition（ 分割 ） 部分，例如第一个 IDE 插槽的硬盘分割成为五个分割槽，此时，您必需要了解什么是 Primary 与 Extended 及 Logical 等磁盘分割槽的概念才行！必需清楚的知道一个硬盘最多只能有一个 Extended ，而 Primary + Extended 最多只能有四个！至于 Logical 则是由 Extended 再分割出来的。

每一颗硬盘分割出来的 partition 代号为 /dev/hd[a-d][1-63] ，亦即原本的硬盘代号后面加上一组数字就是了。 举例来说， IDE1 的 Master 的第一个 partition 即是： /dev/hda1 ， 由于 1-4 号是保留给 Primary 与 Extended 分割槽的，因此，第一个由 Extended 分割出来的 Logical 代号则为 /dev/hda5（注意：这里举例的是第一个 IDE 的 Master 硬盘接法 ）。这些部分在『鸟哥的 Linux 私房菜 -- 基础学习篇』有详细的说明，如果您还是不了解，可得赶紧回去翻一翻啊！因为这很重要的！

还不止此呐，上面这些动作的判断尚包含了： Linux 系统档案的树状目录、 档案所在目录的搜寻方法、 磁盘配额 （ Quota ） 的设定、 文书编辑器 vi 的使用、 核心的编译技巧、 硬盘的安装与维护等等，所以啰， 基础的文件还是需要阅读过的呐！

Linux 安装硬盘规划

OK！了解了架站目的，也知道硬件应该是如何搭配之后，接下来，自然就是安装 Linux distribution 了！到底选择哪一套 distribution 好呢？是 Fedora/SuSE/Mandriva 还是.....这个时候您就必需要清楚的了解到 Linux 其实就是一个『Kernel』啰！而目前 (2006/02) 的 Kernel 最常用的就是 2.6 这个版本，如果您使用较旧的 Linux distribution 例如 Red Hat 7.x 时，那么 Kernel 版本是较为早期的 2.4 版，这个时候，在新版的 2.6 核心上面开发出来的各种类软件就无法在 Red Hat 7.x 的系统上面动作了， 而且，不同核心的函式库也不相同啊！ 所以说，选择 Linux distribution 时，必需要知道该 distribution 的核心版本才好。

一般来说，我们会建议大家不要使用太冷门的 distribution ，因为，支持度可能会比较不够好！所以，目前较为推荐的还是台湾地区比较多人使用的 Red Hat 系统 (Red Hat/Fedora/CentOS 均是 Red Hat 系统)，以及操作接口良好的 SuSE ， 还有 Mandriva 也不错啊！

选择了 distribution 后，当然要开始安装了！安装的第一个要件就是刚刚上个步骤的规划， 因为上面提

过了，所以这里我们就不再谈规划的工作了。规划完成之后，再来就是整个安装的流程了。 安装流程最重要的大概有三个地方：

1. 硬盘的 partition 与挂载：

   除了硬盘的代号需要特别留意之外，当然磁盘分割也不能不清楚啊！磁盘分割方面，需要学习的有 fdisk 这个重要的分割程序。 在分割完成之后，接下来就需要格式化硬盘啦！格式化就需要了解 mke2fs 这个指令的用法了！ 再来，格式化完成后，就需要与挂载点 (目录咯！) 搭配来挂载！挂载使用的指令为 mount。 而挂载前，想要检查一下该 partition 有没有问题，就需要使用 fsck 来检测，另外，记得啊！使用 fsck 时，要被检查的 partition 请务必给他卸载啊(使用 umount)。

   还有还有，如果要让某些 partition 在开机的时候就自动挂载，要记得写入 /etc/fstab 当中，或者是将指令完整的写到 /etc/rc.d/rc.local 当中。哇！这么多指令怎么看的懂？！呵呵！这时又得要知道 man 这个好用的指令的！ 这部份硬盘的管理真的很重要的！不要忽略了！
2. 套件的选择与安装：

   好了，将硬盘整理好之后，终于要将 Linux 安装上来了！这个时候请特别留意，因为 Linux 提供了图形接口与文字接口的预设登入 (run level)，事实上，图形接口挺容易造成系统的资源损耗，因此服务器上面较少使用图形接口喔！ 所以，通常鸟哥都是不安装图形接口的套件的啦！再来，为了未来的升级与重新安装套件的便利性， 所以，在选择套件时，请务必将底下几个套件选择进来：

   - make
   - gcc
   - kernel-source 及/或 kernel-header

   因为 make 与 gcc 是编译套件所必需要的软件，至于 kernel source 或 kernel header 则是一些驱动程序在编译时会使用到的一些函式库或系统数据， 这些东西对于桌上型计算机并不很重要，因此在各大 distribution 的预设套件上面『都没有安装』， 所以，您必需要自行挑选啊！这些套件通常都会放在 Software develop 或者 Utility 的项目当中，请仔细的选择喔！

3. 开机的设定 (Grub/Lilo) ：

   在我们进入系统之前，主机会先读取 BIOS 的信息，然后会读取第一块硬盘的主要开机扇区 (Master Boot Recorder, MBR)，这个动作是为了让我们的主机了解数据格式， 以顺利的将系统的数据读取进来啊！而在这个 MBR 上面的程序，就被称为开机管理程序了！在 Linux 上面主要的开机管理程序有 Grub 与 Lilo 这两支，他们的运作方式有点不太相同！ LILO 是直接将所有的信息都给他写入 MBR 里面，所以如果您设定完成设定档 /etc/lilo.conf 之后，还得将 LILO 重新安装到 MBR 当中。至于 Grub 则是使用类似指向 (point) 的功能，将开机信息导向设定文件 /boot/grub/menu.lst 当中！ 这两个管理程序各有优缺点，可依照个人喜好来选择安装。

   另外，其实我们的主机是可以达成所谓的『多重开机』的系统的！也就是一部主机上面可以有多

个操作系统，包括 Windows 与多个 Linux 。如果要达成多个系统在同一部主机上面的话，您又得要必须了解 MBR 与 Super Block 的异同点才行！这都是需要学习的呢！

真的很不容易喔！安装一部好的 Linux 主机，最大的重点就在于硬盘的 partition 了！硬盘分割的考虑会影响到您的主机未来的扩充性与实用性，还有『安全性！』呢！所有的种种都需要有一定程度的 Linux 概念才行呐！

了解欲架设的网站服务原理

事实上，了解每种服务的运作原理，对于您未来在进行除错 (debug) 是相当的有用的啊！而且，在主机的规划上面也会有一定程度的帮助。举例来说好了，在 Linux 上面很有名气的档案服务器 SAMBA ，他的运作原理主要是 NetBIOS over TCP/IP ，而如果您了解最原始的 NetBIOS 是无法跨网域的，亦即无法跨路由器 (Router) 的，那么就比较容易了解为何 Windows 的网络上的芳邻显示的计算机数量只有局域网络内部这么多而已！

另外，如果您熟悉 FTP 的运作模式时，才有可能会了解『被动式』与『主动式』联机对于 FTP 主机的设定其实是『大有关系』的！或许在这里您完全不晓得鸟哥在谈些什么 (因为这些基础知识在后续的章节才会陆续的提及啊！这里还没有讲到啦！) ，只不过，请大家先有个概念，理解服务 (Services) 的运作流程，将有助于您未来的架设与维护喔！所以，这一部份也不要忽略了！

这还只是各个服务器的服务原理呢！如果是在 Linux 上面运作呢？！那么您至少就得要了解『什么是 daemon ？』而 daemons 的形式有所谓的 stand alone 以及 super daemon 的管理！这两种形式有什么差别？！每种 daemon 管控的 port number 是否相同？如果需要更改 daemon 的 port number ，应该要改 /etc/services 这个档案，您是否了解如何去修改？还有还有，除了服务器才需要的 daemon 之外，其实我们 Linux 主机里面本来就有很多 daemon 存在，例如几乎一定要存在的 crond, syslogd, atd 等等，这都是需要了解的基本知识！

服务的套件安装、 漏洞修补、套件升级...

好了，假设您已经将网络硬件配置搞定了，主机也规划好了，并且也已经可以连上 Internet 了，此外，也已经知道了该服务器的服务原理，那么再来当然就是：『 我的主机上面是否有我要架设的服务器软件了？』举个例子， 如果我问你，你的 Linux 主机上面有没有 Apache 这个 WWW 服务器的软件呢？！聪明的管理员大概已经想到了使用 RPM 来寻找，而如果不是使用 RPM 来管理软件的系统管理员，也会立刻想到 locate, find, which 等等的指令来搜寻相关的档案或指令，例如 Apache 的主要设定档是 httpd.conf ，那么只要找到该档案，就能够了解是否已经安装了这个套件～因此，立刻使用 locate httpd.conf 即可发现啦！

好了，假设您的主机并没有安装 Apache 这个套件，那么您要如何安装呢？ 这个时候就需要考虑到『套件管理员』这咚咚了！目前 Linux 上面的套件管理员大多使用 RPM 与 Tarball 这两个咚咚！使用 RPM 最大的优点是方便管理！因为所有的档案与信息都有纪录，所以在搜寻、升级、反安装上面都相当的容易！ 不过，却也因为相依属性的问题常常导致新手无法立刻进入状况！此外，RPM 有版本方面的问题，不同的 Linux distribution 上面的 RPM 还不能互相挪用呐！呵呵！真是苦恼～

如果您对于 RPM 的相依属性有相当大的反感程度，那么使用原始码 (source code) 来进行编译则是一个不错的思考方向。在系统上面能够执行的档案属于 Binary(二进制文件) ，那么这些 binary 是怎么来的呢？！我们以 Linux 常见的 binary 制作方法：使用 gcc 来编译的动作来说明好了。要制作 binary file 之前，首先就必须要撰写程序代码，这些程序代码大多是以文书编辑器编辑而成的 ASCII 格式档案，这就是通称的『Source code』咯。然后这些程序代码必需要经过编译器 (compiler) 编译成为我们的系统认识的 binary 可执行档才行！在编译的过程中，可能还会使用到很多的函式库 (library) 呢！需要注意的是，目前 linux 上常见的编译器就有 gcc，g77 等，您的原始码必须要针对这些编译器的语法进行撰写才行！

一般来说，当套件释出时，大多采用原始码的方式释出的，但是因为原始码所占用的档案空间比较大， 因此常常会加以压缩之后，才放上网站上供人下载，那就是所谓的 Tarball 了！因此，您必须要了解 tar，gzip，bzip，compress 等指令的用法才能够解开 Tarball 的档案！此外，由于 Tarball 是原始码，因此您还必须要有 compiler 以及相关的 make 与 Kernel 相关的函式库，才能够成功的将这个套件给他编译成为可以在您的系统上面跑的 binary file ！使用 Tarball 的安装方式，最大的优点是具有弹性！ 您可以将套件安装在任何您想安装的目录，还可以自行加入一些额外的参数来设定该套件呢！ 不过，还是有缺点的啦！那就是当平台不一样时，可能由于某些函式库无法找到， 或者是使用者的基本知识不足，就无法成功的将 Tarball 编译成功！另外，用 Tarball 安装时，某些特殊的套件很难进行反安装的动作呐！ 造成升级与移除上的困扰！

反正 RPM 与 Tarball 是各有优缺点啦，鸟哥个人比较偏向于使用系统预设的 RPM 来进行服务器的设定，不过，由于并非每个 Linux distributions 都适合某个套件的 RPM 参数，所以，这个时候我就会开始考虑使用 Tarball 了！无论如何， 既然您要架设服务器，就必须要有该服务器的套件在您的系统上面，那么学习上面这两个套件管理员，是必要的动作！

漏洞修补的重要性：很多的新手在架设好了服务器之后， 就以为『万事 OK 』了！ 所以就不再继续的监视网络上面公布的套件漏洞信息！事实上这是很危险的！因为目前由于 Internet 的发达，网络的危险性其实是越来越严重的！稍一不小心，您的主机可能就会立刻的『中标』。 为了随时修补漏洞，您应该要熟悉如何进行套件的升级！一般来说，使用 RPM 安装的套件就以 RPM 的方式来升级，使用 Tarball 的话，比较麻烦， 需要先移除后再进行升级！无论如何，套件的升级是系统管理员经常要进行的工作！

Tips:
程序设计师所撰写的程序并非十全十美的，所以，总是可能有些地方没有设计好， 因此就造成所谓的『程序漏洞』啰。程序漏洞所造成的问题有大有小， 小问题可能是造成主机的当机，大问题则可能造成主机的机密数据外流，或者主机的操控权被 cracker 取得。在现今网络发达的年代， 程序的漏洞问题是造成主机被攻击、入侵的最主要因素之一了。 因此，快速、有效的针对程序漏洞进行修补，是一个很重要的维护课题。



主机设定、启动、观察与除错

在所有的前置作业都完成之后，终于可以来到『主机设定』的地方了！ 所以您看看，要设定一部堪称完美的主机，前置作业就得学会这么多的基础功夫呐！ 并不简单喔！好好的用功学习吧！主机的设定大致的流程是这样的：

1. 找出设定档：

   主机设定第一个步骤就是需要『找到主要设定档』，因为不论您使用的是 RPM 还是 Tarball， 由于都是同一个套件，所以设定档的档名是不变的！举例来说， Apache 的设定档档名都是 httpd.conf ，而 SAMBA 的设定档档名都是 smb.conf ，您必须找到该设定档之后才能够进行设定啊！所以，熟悉 locate, rpm, find, grep 等指令就显的很重要了！

2. 编辑设定档：

   既然要设定，当然就需要编辑啦！既然要编辑，那么 Unix Like 标准的文书处理器 vi 是否学会了呢？！ vi 是学 Linux 过程中相当重要的一课！如果不会 vi ，那么学习 Linux 之路就会显的跌跌撞撞呢！此外，设定档的内容该如何编辑呢？例如 httpd.conf 里面有些虚拟主机的设定项目，要如何设定呐？！要了解这里面的设定项目，您就必须要学会使用 man, info 等指令，也需要知道套件的文件数据 (documentation) 放置在我们系统的 /usr/share/doc 里面说！如果您知道如何快速的查阅设定项目，那么设定文件的编辑对您而言， 就简单的很呐！

3. 启动服务器：

   设定完成服务器的设定档之后，再来就是需要启动服务器啦！ 而且，如果您在服务器启动之后进行设定档的修改，也需要重新启动服务器才行喔！ 要启动服务器，您就必须要了解什么是 daemons ，而 daemons 又有 super daemon, stand alone 两种模式，在 Linux 预设的路径当中，stand alone 的服务在 /etc/rc.d/init.d/* 这个目录当中，而里面的档案是以 BASH shell script 写成的，所以除了了解 daemon 之外，您还必须要熟悉 shell script 的相关语法才行！另外，如果是 super daemon 的话，必须知道启动服务器的设定档会放置 /etc/xinetd.d 里面，启动的话，则是重新启动 /etc/rc.d/init.d/xinetd 这个 super daemon 才行！

   万一您的服务器软件是以 Tarball 安装的，那么启动的时候可能是直接执行 binary file ，如此一来，就没有 shell script 帮助您启动、关闭、 重新读取设定档等服务器启动的动作！当真如此的话，您就必须要以 程序 (process) 与讯号 (signal) 的方法使服务器动作了！这部份您就必须要熟悉 ps, top, kill 以及 signal number 的意义等等！很重要的呐，尤其是当您想要将目前某个联机中断时， netstat 配合 kill 的用法是很重要的。

4. 观察启动的状态：

   虽然似乎已经启动了服务器，但是启动后的服务器就一定能够正常的运作吗？ 如何观察他是否正常的运作呢！？首先，您就必须要具有 PID 的概念，利用 netstat 观察 PID 与 port number 的讯息，来观察服务器是否正确的在工作呢？！还有，任何系统信息都会 记录到登录文件 (log files) 里面去，所以，启动完服务器后，到该服务器的登录档当中察看一下信息，是相当正确的一个行为！例如启动 DNS 之后，虽然观察 port 确定有启动，但是其实服务器可能是不正常的启动，此时就必须要观察 /var/log/messages 的内容来判定 DNS 的设定是否正确说～ 而既然要观察登录文件，那么 linux 主机上面控制登录文件的 syslogd 这个 daemon 就不能不知道啊！您必须要了解 syslogd 的设定档在 /etc/syslog.conf ，并且可以搭配 logrotate 来进行登录档的轮替！

5. Server 与 Client 的权限问题：

   好了，服务器已经正常的启动了！观察所有的状态也都没有问题！那么我总可以对 Client 进行正确的服务了吧？！『错！』您还尚未考虑到『权限』的概念呐！举个例子来说，架设过 WWW 主

机的朋友大概都知道如果需要开放个人家目录的首页时，将该使用者的家目录设定权限为 755 是必须的！因为如此一来，启动 Apache 程序 owner 才能够进入该目录进行浏览的动作！ 为了要了解权限的概念，您必须至少具有 UID, GID 等 Linux 系统上面的账号概念， 而每个账号的特殊参数在 /etc/passwd 与 /etc/shadow 也是必须要知道的！此外，每个档案或目录具有十个属性的特征也是最基础的观念， 真的重要呐这一部分！而如果要让使用者管理系统的话，身份转换成为系统管理员 (root) 也是必须的！如此您就必须要教育使用者了解 su 及 sudo 的用法！再来，为了预防系统被破坏，适时的减少 SUID 与 SGID 等特殊权限的 binary file 则显的相当的重要！咦！这些东西都不懂～不要架站喔！ ^_^

6. 设定开机启动该服务器：

   终于将服务器设定好，启动正确，对于 Client 端的权限与服务也设定妥当， 再来就是要进行一开机就将服务器加载内存的动作了。如何进行这个动作呢？在 Red Hat/SuSE/Mandriva 有 chkconfig 可以辅助， Red Hat 更有 ntsysv 可以达成简单的设定！然而，如果是其它的 Linux distributions 呢？那么就得了解正常的开关机程序，这里面包含了 Run Level 的观念，Run Level 的读取档案在 /etc/inittab 里头呢！还得要了解 /etc/rc.d/ 里面的目录与 /etc/rc.d/rc.local 这个档案的用途才行！一般来说，我们使用 Tarball 的套件想要在开机时就启动，都会藉由 /etc/rc.d/rc.local 这个档案来达成！呵呵！很重要吧！

经过上面的流程，您就可以知道啦，架设好一部主机需要知道：(1)各个 process 与 signal 的观念；(2)账号与群组的观念与相关性；(3)档案与目录的权限，这当然包含与账号相关的特性； (4)套件管理员的学习；(5)BASH 的语法与 shell scripts 的语法，还有那个很重要的 vi 啰！：(6)开机的流程分析，以及记录登录文件的设定与分析；(7)还得知道类似 quota 以及连结档等等的概念。要知道的真的很多，而且还是不能省略的步骤喔！

客户端设定、观察与除错

一般来说，目前的服务器大多只要针对 Server 设定好即可， Client 端我们不太需要去管理的。不过，某些特殊的套件，例如 SSH, Mail, SAMBA, NAT 等等，就必须要连同 client 的权限与设定一起包含进去设定呢！ 如果您是一个系统管理员的话，那么『教育 Client 端的使用者，正确的使用网络与主机提供的服务』就是一个相当重要的工作了！

最常发生的错误在于 Client 端架设了『个人防火墙』，这部份相当的恼人～那我怎么知道 Client 端的服务要求被防火墙给他中断呢？这当然可以由 Server 以 netstat 来简略的检查，当然，到 Client 端视查一下使用者的使用习性与操作系统，也是一个可行的方案。不过，总体来说， 教育您的 Client 使用者具有最最基础的 Linux 账号、群组、档案权限等概念，才是一个彻底解决问题的方法说！尤其是 Client 端的使用者在使用类似 SAMBA (网络芳邻) 进行数据存取时，最容易发生权限观念这样的错误了！

总之，系统管理员对于 Client 端的使用者还是有一定程度的责任与义务的，至少我们要进行好教育的任务！

安全性设定

前面说过，网络的安全有越来越需要注意的趋势。所以，架设一个相对安全的网站是很重要的！ 那么如何架设好一个相对安全的网站呢？您至少需要有这样的概念：

1. 严格规范使用者的密码设定规则：
   『猜密码』仍是一个不可忽视的入侵手段！例如 SSH 如果对 Internet 开放的话，您又没有将 root 的登入权限关闭，那么对方将可能以 root 尝试登入您的 Linux 主机， 这个时候对方最重要的步骤就是猜出您 root 的密码了！如果您 root 的密码设定成『1234567』哈哈！想不被入侵都很难～ 所以当然需要严格的规范使用者密码的设定了！那么如何规范严格的密码规则呢？可以藉由 (1)修改 /etc/login.defs 档案里面的规则，以让使用者需要每半年更改一次密码，且密码长度需要长于 8 个字符呢！(2)利用 /etc/security/limits.conf 来规范每个使用者的相关权限，让您的 Linux 可以较为安全一点点～(3)利用 pam 模块来额外的进行密码的验证工作。
2. 利用 Super daemon 与 TCP Wrappers 管理服务权限：
   如果您使用 xinetd 这个 super daemon ，或者是直接使用 tcp wrappers 的函式库，那么您将可以直接使用 /etc/hosts.allow 以及 /etc/hosts.deny 来管理是否能够登入系统的某个 daemon 的权限！在 hosts.allow(deny) 里面，能够限制的有 IP, 网段, 网域等等的设定，如此一来，可以让您的 daemon 提供有限的信任网域，毕竟是安全一些的！
3. 利用 netfilter 防火墙：
   除了 /etc/hosts.allow(deny) 之外，利用防火墙机制 iptables 来设定您的主机单机防火墙是很重要的！ 如果您的核心是 2.4.xx 以上版本(包含 2.6)的话(利用 uname 查询核心版本)，防火墙机制为 iptables ，如果是 2.2.xx 的话，则是使用 ipchains 。如果不是因为特殊需求的话，目前我们大概都会建议大家使用 iptables 这个机制，因为不但设定较为简单，而且功能更为强大！
4. 持续进行套件修补：
   让我们做一个简单的想法：『架设了防火墙，是否就安全无虞了』？ 如果您的答案为『是』，那么仔细的观察一下底下的说法吧！如果您的主机有开放 WWW 也就是 port 80 ，既然要开放 port 80 ，当然防火墙就必须要开放 Client 登入。也就是说，虽然您架设了防火墙，可以抵挡非 port 80 以外的联机，不过， 不论来自何方的联机，只要连接到您的 port 80 ，那么该联机就会予以通过！好啦，万一这个 www 软件被侦测出有漏洞的话，由于您的 port 80 是允许大家登入的，结果呢？呵呵！当然别人就可以利用 WWW 的漏洞成功的入侵您的主机了！这样说可以理解为什么系统管理员需要常常更新套件以修补漏洞了吧？！ ^_^

无论如何，以现今的网络功能及维护来看， 架设一个『功能性强』的主机，还不如架设一个『稳定且安全的主机』比较好一点！ 因此，对于主机的安全要求就需要严格的要求啦！就鸟哥的观点来看， 如果您的主机是用来替你赚钱的，例如某些研究单位的大型 Cluster 运算主机，那么即使架设一个甚至让您觉得很不方便的防火墙系统，都是合理的手段！ 因为主机被入侵就算了，数据被窃取，呵呵！那可不是闹着玩的！

服务日志、登录文件与备份管理

除了安全性之外，主机也可能因为硬件问题或者是人为使用不当而产生错误讯息等问题！这些讯息会放置在 /var/log 里面，不过，还是得视 syslogd 这个 daemon 的设定档 /etc/syslog.conf 的设定而定喔！这些登录文件的信息是相当重要的，他可以记录曾经发生过的事情， 如果再经由系统管理原自行写的分析软件，那么就可以很轻松的管理好主机了！ 也可以在最短的时间内发现主机的可能漏洞呢！不可说不重要啊！

我们常常说主机的服务越单纯越好，原因是什么呢？如果哪一天我们发现主机的登录文件有点问题， 要来进行入侵管道的查询，万一主机的服务过多，很难追查出到底是哪一个 daemon 造成主机的问题啊！如果主机的服务很单纯，分析登录档也会比较轻松呢！ 此外，系统管理员最好有制作工作日志的习惯，可以让您未来管理主机比较容易快速的进入状况！ 总之，要良好的管理主机，利用主机的工作排程(crontab)也是很重要的喔！

当然啦，主机随时的给他备份是一个很良好的行为，要怎么备份呢？！ 您必须要了解主机的相关信息，例如邮件主机，您可能就得备份 /etc 与 /home 及 /var/spool/mail 等目录，而如果是 WWW 主机，就得找到 WWW 主页的目录，才能进行备份吶！要用什么工具呢？可以选择 tar 或者是 cpio 等工具，当然，您还得配合备份的媒体，例如抽取式硬盘啊、可烧录光盘啊等等的媒体说！

小结语

由上面的整个架站流程来看，由规划到安装、主机设定、账号与档案权限管理、 后续安全性维护与管理以及重要的备份工作等等，必需要每个环节都很清楚， 才能够设定出一个较为稳定而可正常工作的服务器。而上面的每一个工作都涉及到相当多的 Linux 基础操作与相关的概念，所以说， 想要学架站，真的真的不能省略了 Linux 的基础学习，这也是为什么我们一再强调 Linux 新手不要一头栽入想要单纯架设服务器的迷思当中吶！ 如果您对于上面谈到的几个基础概念不是很清楚的话，那么建议您由底下的两个网站学起：

- http://www.study-area.org
- http://linux.vbird.org

在这一本书当中，鸟哥并没有再花篇幅再继续介绍一些 Linux 的基础指令以及相关的设定， 这一本书本来就定位在已经具有 Linux 知识的朋友的工具书，所以，对于尚未接触过 Linux 的朋友来说，这部份其实并不适合您！建议您还是得要重头学起吶！若想要一本 Linux 基础知识的工具书在手边，也可以考虑鸟哥前一本着作 『鸟哥的 Linux 私房菜 -- 基础学习篇』喔！ (好像有点老王卖瓜了喔！ ^_^)大家加油的啦！

自我评估是否已经具有架站的能力

由前一节的内容介绍中，不难知道要架设一个堪称完美的网站， 得事前花诺大的心力来培养您自己的『能力』才行！说实在的，架站实在不能贪图『便利』， 因为越是『便利自己』就越是『便利 Cracker 的入侵』啊！此外，上面谈到的都属于技术层面的部分喔！ 要有架站的能力，不只有『技术』就可以了喔！还需要其它心理层面的辅助呢！ @_@ 咦！搞什么～架站还需要心理层面的因素？我们是在玩心理战是吧？！

这里要请您特别注意的一点是，如果您架了一个网站，一般来说，您自然就是系统管理员 (root) 的身份了。而要晓得的是，在 Linux 系统当中， root 具有『至高无上』的权力， 他可以进行任何系统的设定，也可以察看任何使用者的档案或邮件或什么机密文件等等的， 还可以让使用者寄出、寄入的邮件都送一份到自己的信箱去！所以，如果身为 root ，还拥有『极高度的偷窥欲望』时，那么您的 user 岂不是毫无秘密可言？换个角度来想，如果您是一般的 user ，而您的 root 却拥有很高的偷窥欲，我想，您应该也不见得会有多快乐吧？ 就好像在家里被人家装了针孔式摄影机一样，感觉绝对不可能太好！所以，要成为 root 之前，您必须做好心理建设，那就是拥有相当高水平的『道德感』。

一般来说，网管人员需要什么能力呢？我想， 架几个站跟作一个称职的网管人员，相差是甚远的！架站，说真的，是一件很简单的事情， 看着书本一步一步的作上去，一定可以成功的！但是，很多人都只晓得 『如何架站』却不知到 『如何维护一个网站的安全』！ 基本上， 维护一个已经架设好的网站的正常运作，真的要比架设一个网站难的多了！您得要随时知道您的系统状况， 随时注意是否有新的套件漏洞而去修补他，随时要注意各种服务的登录档案(logfile)以了解系统的运作情况！ 得知道发生问题的时候，到底问题点是在哪一个！比如说当机了，那么您知道当机的原因吗？ 即使不知道，也可得需要约略猜得出来才行。而，如果安全出了问题，被入侵了，除了 format＋ 重灌之外，可有办法在不移除系统的情况下修补漏洞？ 这些都是网管人员需要学习的，而且，通常都是需要经验的累积才会知道问题的所在！ 此外，保持身心的活力以随时注意在线公布的安全防备信息等等！都需要具备的！

此外，最严重的问题是，网管人员其实最需要的是 『道德感与责任感』！您可要晓得您的机器上所有人的隐私都在您的监控之下， 如果您本身就已经有偷窥欲了，可知道这有多可怕吗？！另外，如果没有责任感的人作为一个网管， 可能会疯掉，因为不论何时何地，只要是你监控的主机出了问题，嘿嘿嘿嘿，你一定是第一个被想到的人物， 所以，你得随时随地做好可能随时会被召唤回主机跟前的心理准备！更可笑的是，如果你服务的人群中， 有几个连开机的时候软盘机塞了一块不可开机的软盘，导致无法正常开机， 也都会跟你抱怨说『唉呦！您经手的计算机怎么这么烂，动不动就不能开机』的时候， 您得要有容人的雅量，说说冷笑话解解闷吧！总之，网管人员并不是只要会架站就可以了， 『道德感』『责任感』还有『耐心』呵呵！套一句现在人喜欢说的口头禅『这是一定要的啦！』

那么网管人员是什么？ 前一阵子看到了报纸的一篇报导， 内容大概是说：台湾的网络管理人员对于『网络安全性防护』的认知不够，或许是防火墙机制建立不完整， 或者是认为黑客不会入侵小型网站，所以在不甚了解的情况下，被所谓的『中东黑客组织』所入侵， 然后以台湾被入侵的计算机为跳板，去攻击宾拉登的仇敌美国，然后引起美国高度的不满。由于台湾的立场有点得罪不得美国 (这边不提及政治因素，反正目前的情况是这样。) ，所以一接到美国来的抗议信函就很棘手。 这只是一个事件问题，不过这个事件问题也点出了一个重点，就是我们的网络信息可能真的是蛮发达的， 不过，管理网络的人员可能在认知的程度上就有点参差不齐了！网络安全是蛮重要的，只是， 大家常常会忘记他！个人认为，网管是蛮重要的角色，应该不能等闲视之才对。

好了，如果您了解了上面小弟所想要表达的意念之后，来评估看看您是否适合当一个称职的网管人员吧！

1. 是否具有 Linux 的基础概念：
   这当然包含很多部分，例如账号管理、BASH、权限的概念、Process 与 signal 的概念、简易的硬件与 Linux 相关性 (如 mount)的认识、登录档案的解析、daemon 的认识等等，都需要有一定程度的了解；
2. 是否具备基础网络知识：
   没有网络知识想要架站，那是天方夜谭！请确认您已经熟悉 IP,Netmask,route,DNS,daemon 与 port, TCP 封包的概念等基本知识；
3. 是否已经身心活化了：
   网管人员必须要随时注意网站的相关信息，这包括网站套件的漏洞修补、 网络上公告的网络安全通报等等，还有，得要每日分析主机的登录文件， 您是否已经具备了随时注意这些信息的『耐心』呢？
4. 是否具有道德感与责任感：
   如果还是具有一点点的偷窥欲，再加油吧！^_^ ，另外，如果老板想要请您『偷窥』时，请想尽任何方法，让他理解这么做是多么的可笑～

当然，一再强调的，架设一个 Linux 服务器是很简单的，但是维护的工作除了身心已经活化， 并且还要拥有高标准的道德感，否则.....倒站可能是可以预见的一个后果.....

课后练习：

- 请简述进行网站架设前，应该具备何种基本技能？
- 如果我有一颗硬盘在 A 主机上面安装了 Linux 之后，拿到另一台配备相同的 B 主机上面去进行开机，结果竟然无法顺利开机，您认为可能的原因是什么？
- 一般来说，在 Linux 系统上，使用者预设的家目录在那个目录下？另外，新增一个使用者时， 该使用者预设的家目录内容来自那个目录下？
- 磁盘配额 (quota) 能否针对某个特定的目录进行限制？Quota 有什么较为特殊的使用限制？
- 在 Linux 系统下，要寻找一个档名为 vbird.document 的档案，可以使用什么指令进行搜寻？又，如果要寻找在一天内更动过的档案，又该如何进行？
- 在 Linux 系统中，常见的套件管理员有 RPM 与 Tarball ，请分别说明这两个套件管理员的优缺点。
- 如果我下载了一个档名为 httpd-2.0.52.tar.gz 的档案，一般来说，这个档案代表的意义为何？我该如何让这个档案能够在我的 Linux 系统上面安装？
- 我以原始码的方式进行一个套件的安装，但是在分析系统的时候，分析程序一直告诉我找不到 cc 这个指令，请问这是什么问题？为何需要 cc ？又，我该如何解决这个问题，好让套件可以顺利的被安装在我的 Linux 上面？
- 我发现我的 Linux 系统怪怪的，似乎有什么不知名的程序在内存当中跑，我该如何将这个不知名的程序捉出来，并且将他移除？
- 我总是无法编辑某个档案，您认为应该是什么问题造成的？那又要怎么解决？
- 什么是 UID 与 GID ？UID 有哪些等级？
- 使用者的家目录参数、UID、GID 以及其它相关参数，还有密码档案，放置在哪些档案里面？
- 您认为一个称职的网管人员应该具备什么能力？
- 我要启动一个系统预设的 Service，请问我可能可以由执行或修改哪些目录底下的档案来启动？
- 我要关掉 cron 这个服务，应该怎么关掉他？如果正常的方法无法关闭这个服务，可以使用什么方法来关闭？
- 如果一开机就要执行某个程序，应该要将该程序写入那个档案里面？

网络基础概念

最近更新日期：2006/07/16

在金庸的小说里面提到了『欲练神功，挥刀自宫』才能练成无敌的葵花宝典， 另外，金大侠也提到太极拳的学习到最后需要将所学的忘光光，此时才能『无招胜有招』。 呵呵～这跟 Linux 有啥关系呐？在前一章我们不是提到关于『架站前的技巧分析』吗？里面提到在 Linux 里面想要玩架站， 最重要的是得要搞熟 Linux 相关的操作技巧，这些技巧在一开始时真的得要花很多时间去熟悉， 真的会让人觉得好像已经『自宫』的样子！不过，别担心， 等您花时间学习过后，肯定让您忘不了了！因为那时您已经『无招胜有招』啦！ 发生问题时，肯定会依循之前学习到的一贯解决方法去搜寻、解答！很重要的！

在这一章里面，我们得要来谈的就是另一个『神功』啦！那就是网络基础。 网络基础真的很重要的，包括以太网络硬件的了解可以让您知道如何查出有问题的地方，进一步解决他； 了解网络协议(IP)与路由(route)让您可以完整的设定好您的网络架构，更可进一步进行子网络的划分， 以建立更小而美的网络环境！当然还有不可不提的 OSI 网络七层协议，真是重要啊！ 不过，在这一章鸟哥以较为口语的方式来介绍这些基础网络架构，希望能带给朋友们快速了解网络是啥。 当然，想要更了解网络相关功能的话，文末的参考资料可以参考看看喔！ ^_^

网络(Network)

Linux 这个操作系统的优势之一就是在 (Network) 功能啦！这包含他的高稳定度系统资源分配， 以及较为安全的网络防备能力！所以很多人都喜欢拿他来进行网络服务器的架设。 然而，这些网络功能的优点却同时也是缺点，怎么说呢？因为 Linux 网络功能太强了， 所以一旦被 Cracker (网络怪客) 入侵的话，嘿嘿嘿嘿！会遇上什么灾难你可能也想象不到！ 所以啰，上网之前，请先注意：『我的网络做好安全防护了吗？』

再者，相对于窗口操作系统来说， Linux 的设定确实会让使用者花费比较多的心力， 毕竟当初它是一群工程师由于本身兴趣而设计出来的，所以依旧具有相当麻烦的设定工作需要作！ 这个时候，如果没有一些网络基础方面的认知的话，那么在 Linux 上面设定网络或者是进行网络除错将是您一生当中最大的梦魇......。 所以，这里有几件事情要请您好好的来参考一下：

- 熟悉 Linux 操作与 Linux 基础：
  还是要一再地不断强调，如果您确定您只是想要『Linux 能正常运作就好』那一类型的使用者，那么真的不要再往下看了，因为主机后续的维护问题会很大， 倒不如花个小钱，请个专家来帮您搞定即可！而如果您确定您是『想要更了解 Linux 的使用者，并且想要熟悉架设网站』， 那么请不要再在网络上询问：『为何我不能使用 FTP 传送数据到主机上？』 或者是『为何我不能建立个人网页』之类的傻问题，因为这仅仅牵涉到『 档案权限与属性 』的概念而已，而这些概念都是 Linux 基础里头相当重要的功课！此外，由于远程操控 Linux 主机时，几乎都是使用文字界面来工作，所以不懂 bash shell ？ 哈哈！那么想要干嘛都码不可能！
- 花几晚的时间将网络基础看一看：
  这一个章节旨在引导网络新鲜人快速进入网络的世界，所以鸟哥写的比较浅显一些些， 基本上，还有一堆网络硬件与通讯协议并没有被包含在这篇短文里头。 如果您的求知欲已经高过本章节，那么请自行到书局寻找适合您自己的书籍来阅读！ 当然，您也可以在因特网上面找到您所需要的数据。 在本章最后的参考数据可以瞧一瞧呐！
- 随时掌握主机信息：
  这是最麻烦的一点了！因为大家还是常常认为『我的网站这么小，没有人会注意的啦！』唉！ 说过若干次了，就是因为有这种心理存在，我们才会常常听到 『奇怪！我在早上刚安装完毕，怎么下午就无法以 root 的身份登入了！』 请随时注意您主机的信息，好好的爱护他吧！

好了，底下我们就得要来谈一谈一些重要的网络基础概念了，清醒了没？ 赶紧清醒清醒，准备要好好用功啦！ ^_^

什么是网络

我们都知道，网络就是几部计算机主机或者是网络打印机之类的接口设备， 透过网络线或者是无线网络的技术，将这些主机与设备连接起来， 使得数据可以透过网络媒体(网络线以及其它网络卡等硬件)来传输的一种方式。 请您想象一下，如果您家里面只有计算机、打印机、传真机等机器，却没有网络连接这些硬件，那么使用上会不会很麻烦？如果将这个场景移到需要工作的办公室时， 计算机的数据无法使用网络连接到打印机来打印，那是否很伤脑筋呢？ 对吧！光用想的就觉得很麻烦吧！不幸的是，这些麻烦事在 1970 年代以前，确实是存在的啊！

各自为政的『硬件与软件』技术发展

在 1970 年代前后，为了解决这个烦人的数据传输问题，各主要信息相关的公司都在研究独自的网络连接技术， 以使自家的产品可以在办公室的环境底下组织起来。其中比较有名的就是全录公司的 Ethernet 技术， 以及 IBM 研发的 Token-Ring 技术了。但是这些技术有个很大的问题，那就是这些技术彼此不认识对方的网络技术， 也就是说，万一你的办公室购买了整合 Ethernet 技术的计算机主机，但是其它的计算机却是使用 IBM 的机器时， 想要在这两者之间进行数据的沟通，在早期来说那是不可能的。

以『软件』技术将硬件整合

但是，这些硬件的技术出现之后，还是对企业造成一定程度的困扰，怎么说呢？ 因为一个公司不太可能仅会使用一家厂商所推出的信息产品吧！所以啰，这么多的硬件技术又该如何整合呐？ 举例来说， IBM 不可能不用自己的 Token-Ring 技术，当然也不会将该技术用在其它公司的硬件上面， 所以，这些厂商当然只会针对自家的硬件来进行网络传输软件的撰写啰。那么当许多不同公司的产品在自己企业内时， 您该如何将这些咚咚整合在一起呢？伤脑筋是吧！

所以在 1960 年代末期美国国防部就开始研究一个 可以在这些不同的网络硬件上面运作的软件技术， 使得不同公司的计算机或数据可以透过这个软件来达成数据沟通。这个研究由美国国防部尖端研究企画署 (Defense Advanced Research Project Agency, DARPA)负责，他们将该网络系统称为 ARPANET， 这个咚咚就是目前熟知的 TCP/IP 技术的雏形了！在 1975 年左右， ARPANET 已可以在常见的 Ethernet 与 Token-Ring 等硬件平台底下互通数据了。DARPA 在 1980 年正式推出 TCP/IP 技术后， 由于想要推展此项技术，因此与柏克莱 (Berkeley) 大学合作，将 TCP/IP 植入著名的 BSD Unix 系统内， 由于大学乃是未来人才数据库的培养处，所以， TCP/IP 这个技术便吸引越来越多使用者的投入， 而这种连接网络的技术也被称之为 Internet 。

没有任何王法的 Internet

现在我们知道 Internet 就是使用 TCP/IP 的网络连接技术所串联起来的一个网络世界， 而这个 Internet 在 1980 年代之后由于浏览器图形接口的兴起，因此快速的蔓延在计算机世界中。 但是，Internet 有没有人在管理啊？呵呵！很不巧的是， Internet 一个管理相当松散的所在。 只要你能够使用任何支持 TCP/IP 技术的硬件与操作系统，并且实际连接上网络后， 你就进入 Internet 的世界了。在该世界当中，没有任何王法的保护，您的实际数据如果接上 Internet ， 在任何时刻都需要自己保护自己，免得中了『流弹』而受伤啊！

为甚么说 Internet 没有王法呢？这是因为 Internet 仅是提供一个网络的连接接口，所以您只要连接上 Internet 后，全世界都可以任你遨游，不过也因为如此，『跨海』而来的攻击就成了简单的事件， 简单说，台湾的法律仅适用台湾地区对吧？但是计算机怪客可以在国外透过 Internet 对你的主机进行攻击，我们的法律可管不到国外地区啊！虽然可以透过很多国际管道来寻求协助，不过，还是很难协助你缉拿凶手的啊。 因此啰，在你的主机要连上 Internet 之前，请先询问自己，真的有需要连上 Internet 吗？^_^

软硬件标准制定的成功带来的影响

现在我们常常听到『你要上网啊！？那你要去买网络卡喔！还得要连接到 Internet 才行啊！』 这个网络卡就是市面上随处可见的一个适配卡而已，至于 Internet 则是去向 Hinet/Seed net 或 其它网络服务提供公司 (Internet Service Provider, ISP) 申请的账号密码。 问题是，是否就仅有网络卡与 Internet 才能上网啊？呵呵！当然不是！网络的硬件与软件可多着那！ 不过，最成功的却是以太网络 (Ethernet) 与 Internet ，这是为甚么呢？这两者的技术比较好吗？ 当然不是！这是因为这两者都被『标准』所支持的缘故。

以太网络最初是由全录 (Xerox PARC) 所建构出来的，而后透过 DEC, Intel 与 Xerox 合作将以太网络标准化。 再经由 IEEE (Institute of Electrical and Electronic Engineers) 这个国际著名的专业组织利用一个 802 的项目制定出标准，之后有 19 家公司宣布支持 IEEE 所发布的 802.3 标准， 并且到了 1989 年国际标准组织 ISO (International Organization for Standard) 将以太网络编入 IS88023 标准，呵呵！这表示以太网络已经是一项公认的标准接口了，如此一来，大家都可以依据这个标准来设定与开发自己的硬件， 只要硬件符合这个标准，理论上，他就能够加入以太网络的世界，所以，购买以太网络时，仅需要查看这个以太网络卡支持哪些标准就能够知道这个硬件的功能有哪些， 而不必知道这个以太网络卡是由哪家公司所制造的呐。

Tips:
标准真的是个很重要的东西，真要感谢这些维护标准的专业组织。当有公司想要开发新的硬件时， 它可以参考标准组织所发布与维护的文件资料，透过这些文件数据后，该公司就知道要制作的硬件需要符合哪些标准， 同时也知道如何设计这些硬件，让它可以『兼容』于目前的机器，让使用者不会无所适从啊。 包括软件也有标准，早期 Linux 在开发时就是透过了解 POSIX 这个标准来设计核心的， 也使得 Linux 上面可以执行大多数的标准接口软件呢！您说，标准是否真的很重要啊！



当然啦，以太网络的成功除了加入成为国际标准之外，他持续发展成为星型联机也是一个相当重要的影响。之后 Novell 的 NetWare 这个网络操作系统支持以太网络，加上 NetWare 的强大功能与支持 IBM 的个人计算机， 都导致以太网络的流行！直到现在，以太网络是整个办公室与家庭内部的相当重要的一项硬件配备呢！ 他也是等同于基本网络设备的同意字了呐！ ^_^

除了硬件之外，TCP/IP 这个 Internet 的通讯协议也是有标准的，那就是底下的网站所提供的基本文件：

- http://www.rfc-editor.org/

透过这些文件的辅助，任何人只要会写程序语言的话，就有可能发展出自己的 TCP/IP 软件， 并且连接上 Internet 说。早期的 Linux 为了要连接上 Internet ，Linux 团队就自己撰写出 TCP/IP 的程序代码，透过的就是这些基础文件的标准依据啊！举例来说： RFC1122 这个建议文件就指出一些基本需求， 让人们可以了解啊！

透过这些软硬件的标准以及实际上很多公司的支持，让现今的网络世界很容易就串接在一起。 而目前我们最常谈到的就是上面提到的咚咚，硬件就属『以太网络』最为常见， 软件当然就是 TCP/IP 这个 Internet 最通用的通讯协议啰。那么以太网络是啥？ 为甚么有高速以太网络、超高速以太网络？ TCP/IP 是啥？ 通讯协议是啥？浏览器又是啥？他们之间的关系是怎样？这些东西我们就慢慢来了解一下啰。

以太网络

在目前的网络社会当中，常见的网络硬件包括有最常见的以太网络，当然还有速度算是最快的光纤网络，别忘了还有蓝芽无线技术以及 ATM (Asynchronous Transfer Mode, 可不是自动提款机啊！！) 等硬件。会有这么多网络硬件的原因有很多， 只要是将各个网络硬件的使用场合分类吧！举例来说，一般家庭使用的网络速度并不需要太高， 若使用光纤网络，贵的哩！用不起～而企业场合如果仅使用以太网络作为整合接

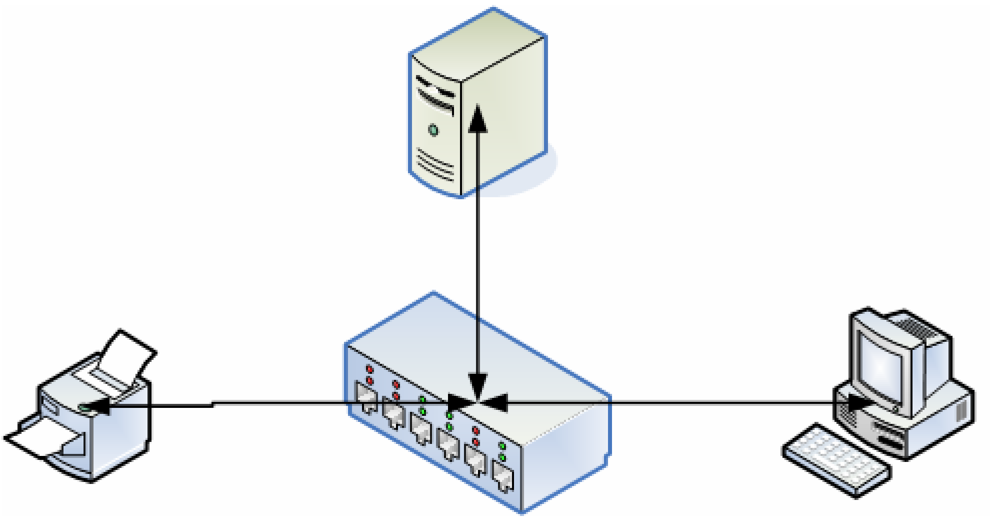
口， 又可能造成频宽的不足！所以啰，这些硬件各有其优缺点啦！

因此，在谈以太网络之前您必须要了解的是，整个网络世界并非仅有以太网络这个硬件接口， 只是由于个人计算机的成功以及相关操作系统的支持度，加上以太网络加入成为标准， 使得以太网络成为目前最为热门的网络硬件技术。 因此，我们当然得就以太网络来聊一聊啦！事实上，整个以太网络的发展建议您可以直接参考风信子与张民人先生翻译的 『Switched & Fast 以太网络』一书，该书内容相当的有趣，挺适合阅读的呐。 底下我们仅做个简单的介绍而已说。

就像前一小节提到的，以太网络最早是由全录这家公司为了自家的硬设备而发展起来的， 经由发明者 Metcalfe 大力推动以太网络成为业界的标准后， 再经由 3Com 发展大量的以太网络硬件，配合越来越流行的兼容于 IBM 的个人计算机， 以及支持网络的操作系统的流行，最后使得大家都参与以太网络这个接口的发展呐， 也由于多数公司的量产，使得以太网络设备越来越便宜。早期的一张 10/100 Mbps 的网络卡要价上千元， 目前最便宜的只要 150 台币就能购买到了！

以太网络的速度
前面说到，以太网络的流行除了相关的硬件以及操作系统的流行之外，『标准』也是一个很重要的因素。 早先 IEEE 所制订的以太网络标准为 802.3 的 IEEE 10BASE5 ， 这个标准主要的定义是：『10 代表传输速度为 10Mbps， BASE 表至采用基频信号来进行传输，至于 5 则是指每个网络节点之间最长可达 500 公尺。』 网络的传输信息就是 0 与 1 啊，因此，数据传输的单位为每秒多少 bit ， 亦即是 M bits/second, Mbps 的意思。那么为何制订成为 10Mbps 呢？ 这是因为早期的网络线压制的方法以及相关的制作方法，还有以太网络卡制作的技术并不是很好， 加上当时的数据传输需求并没有像现在这么高，所以 10Mbps 已经可以符合大多数人的需求了。

当时的网络线使用的是旧式的同轴电缆线，这种线路在现在几乎已经看不到了。 取而代之的是类似传统电话线的双绞线 (Twisted Pair Ethernet) ， IEEE 并将这种线路的以太网络传输方法制订成为 10BASE-T 的标准。 10BASE-T 使用的是 10 Mbps 全速运作且采用无遮蔽式双绞线 (UTP) 的网络线。此外， 10BASE-T 的 UTP 网络线可以使用星形联机(star)， 也就是以一个集线器为中心来串连各网络设备的一个方法， 有点类似底下的图示：



图一、星形联机 (star) 简易图示

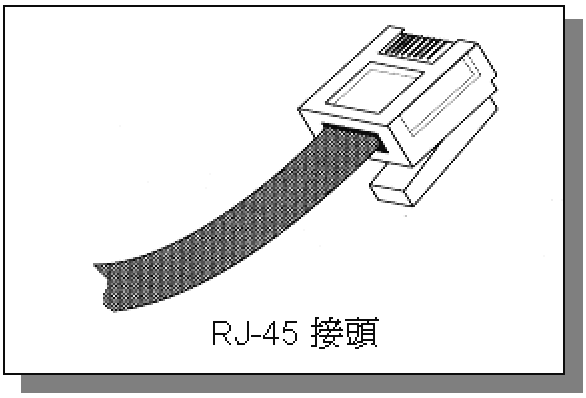
不同于早期以一条同轴电缆线连结所有的计算机的 bus 联机， 透过星形联机的帮助，我们可以很简单的

加装其它的设备或者是移除其它设备， 而不会受到其它装置的影响，这对网络设备的扩充性与除错来说，都是一项相当棒的设计！ 也因此 10BASE-T 让以太网络设备的销售额大幅提升啊！

后来 IEEE 更制订了 802.3u 这个支持到 100Mbps 传输速度的 100BASE-T 标准，这个标准与 10BASE-T 差异不大， 只是双绞线线材制作需要更精良，同时也已经支持使用了四对绞线的网络线了， 也就是目前很常见的八蕊网络线呐！这种网络线我们常称为等级五 (Category 5, CAT5) 的网络线。 这种传输速度的以太网络就被称为 Fast ethernet 。至于目前我们常常听到的 Gigabit 网络速度 1000 Mbps 又是什么呐？那就是 Gigabit ethernet 哩！只是 Gigabit ethernet 的网络线就需要更加的精良。

为什么每当传输速度增加时，网络线的要求就更严格呢？ 这是因为当传输速度增加时，线材的电磁效应相互干扰会增强， 因此在网络线的制作时就得需要特别注意线材的质料以及内部线蕊心之间的缠绕情况配置等， 以使电子流之间的电磁干扰降到最小，才能使传输速度提升到应有的 Gigabit 。 所以说，在以太网络世界当中，如果您想要提升原有的 fast ethernet 到 gigabit ethernet 的话， 除了网络卡 (Network Internet Card, NIC) 需要升级之外，主机与主机之间的网络线， 以及连接主机线路的集线器/交换器等，都必须要提升到可以支持 gigabit 速度等级的设备才行喔！

以太网络的网络线接头
前面提到，网络的速度与线材是有一定程度的相关性的，那么线材的接头又是怎样呢？ 目前在以太网络上最常见到的接头就是 RJ-45 的网络接头，共有八蕊的接头，有点像是胖了的电话线接头， 如下所示：



图二、RJ-45 接头示意图

而 RJ-45 接头又因为每条蕊线的对应不同而分为 568A 与 568B 接头，这两款接头内的蕊线对应如下表：

接头名称	1	2	3	4	5	6	7	8
568A	白绿	绿	白橙	蓝	白蓝	橙	白棕	棕
568B	白橙	橙	白绿	蓝	白蓝	绿	白棕	棕

事实上，虽然目前的以太网络线有八蕊且两两成对，但实际使用的只有 1,2,3,6 蕊而已， 其它的则是某些特殊用途的场合才会使用到。但由于主机与主机的联机以及主机与集线器的联机时， 所使用的网络线脚位定义并不相同，因此由于接头的不同网络线又可分为两种：

- 并行线：两边接头同为 568A 时称为并行线，用在连结主机网络卡与集线器之间的线材；
- 跳线：一边为 568A 一边为 568B 的接头时称为跳线，用在直接连结两部主机的网络卡。

而不同等级的线材除了针对线材材质的电阻等规格加以规范之外，有时为了更好的电磁效应屏蔽功能， 会将四对蕊线以金属薄膜包覆，以提供更佳的抗干扰能力。 没有屏蔽的我们就称为无遮蔽双绞线(UTP)， 有屏蔽的就被称为有遮蔽双绞线(sheild Twisted Pair, STP)。 STP 的网络线由于加上屏蔽物质，所以较硬、较贵也较不易布线，不过优点则是对于电磁效应屏蔽较佳。 那么网络线如何选择？以目前来说，由于我们想到达到 Gigabit Ethernet 的网络速度， 所以必须使用 CAT 5e 以上等级包含 CAT 6 的网络线材才行！ 那么如何区分？ 其实在网络线上面的缆线表面都会写上这条网络线的相关规范，看一看就知道啦！ 而且还得要看看该条线段是『跳线』还是『并行线』喔！

数据在以太网络间的传送 (MAC)

接下来要谈的是那么以太网络到底是如何传输数据的呢？ 由于目前办公室内部的以太网络多是利用集线器以及交换器(Hub/Switch)做为中心， 利用星形联机达成网络环境的一种方式，因此网络线是一个很重要的媒体喔！ 那么网络线里头最多就是电子讯号在跑嘛(就是 0 与 1 啊)！而如果同时有两部计算机要使用这个网络线时， 怎么可能同时发出两个电子讯号出来呢？这个时候是会发生讯号碰撞的问题的， 因此，网络共享媒体(包括网络线、集线器等)在单一时间点内， 仅能被一部主机所使用这个概念必须要先了解才行。

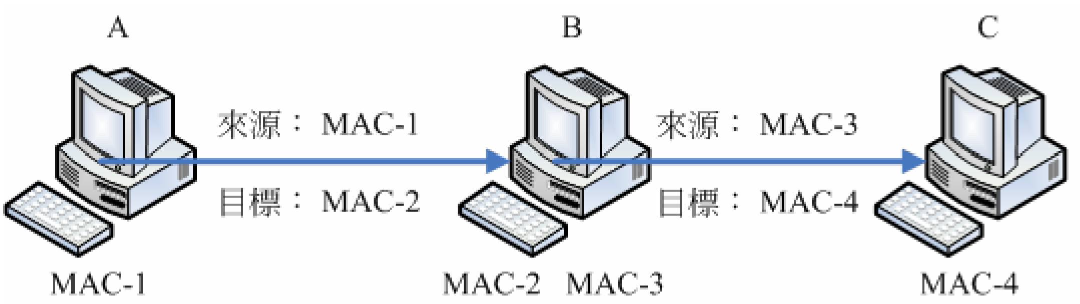
为了杜绝这种讯号碰撞产生的问题，所以以太网络在发展时就使用一种名为 CSMA/CD (Carrier-Sense Multiple Access with Collision Detection) 的技术。 这种技术的特点就是当节点想要在网络媒体上面传输数据时， 会先侦测该媒体上面是否有其它的节点正在使用，确定没有其它节点在使用该媒体时，该节点才会开始传送数据。 并且，当节点开始传送数据时，节点也能够同时侦测是否有发生碰撞的现象。 (注：任何一个具有 MAC 的网络媒体接在该网络上面,就称为一个节点"node", 所以,一部主机就是一个 node !) 不过，事实上在这样的共享媒体环境下，当网络相当忙碌时， 数据的碰撞 (collision) 还是会发生的啦！

再来，我们还是得要知道一下『那电子讯号由一部主机发出后，这个讯号怎么知道要传到哪里去？』 既然有来源也有目标，那当然就得需要一个可以判别讯号来源与等待接受的主机的相关信息啰？ 没错！在以太网络内，我们就是以 MAC (Media Access Control, 媒体存取控制)来管理数据传送的。 而 MAC 其实就是一个讯框(frame)， 你可以把他想成是一个在网络线上面传递的包裹， 而这个包裹是整个网络硬件上面传送数据的最小单位了。 也就是说，网络线可想成是一条『一次仅可通过一个人』的独木桥， 而 MAC 就是在这个独木桥上面动的人啦！那 MAC 又该如何判断这独木桥的两端分别是何处呢？ 这就得要看一看 MAC 这个讯框的内容了：

前導碼 8 Bytes	目的位址 6 Bytes	來源位址 6 Bytes	資料欄位通訊 2 Bytes	主要資料 46-1500 Bytes	檢查碼 4 Bytes

图三、以太网络的 MAC 讯框

在这个 MAC 当中，最重要的就是那个 6 Bytes 的目的与来源地址了！ 事实上，在所有的以太网络卡当中都有一个独一无二的网络卡卡号， 那就是上头的『目的与来源地址』，这个地址是硬件地址(hardware address)， 共有 6 bytes ，分别由 00:00:00:00:00:00 到 FF:FF:FF:FF:FF:FF， 这 6 bytes 当中，前 3bytes 为厂商的代码，后 3bytes 则是该厂商自行设定的装置码了。 在 Linux 当中，你可以使用 ifconfig 这个指令来查阅你的网络卡卡号喔！ 不过，由于 MAC 主要是与网络卡卡号有关，所以我们也常常将 MAC 作为网络卡卡号的代称。 特别注意，在这个 MAC 的传送中，他仅在局域网络内生效， 如果跨过不同的网域 (这个后面 IP 的部分时会介绍)，那么来源与目的的地址就会跟着改变了。 这是因为变成不同网络卡之间的交流了嘛！所以卡号当然不同了！ 如下所示：



图四、在不同主机间持续传送相同数据的 MAC 讯框变化

例如上面的图标，我的数据要由计算机 A 通过 B 后才送达 C ，而 B 计算机有两块网络卡， 其中 MAC-2 与 A 计算机的 MAC-1 互通，至于 MAC-3 则与 C 计算机的 MAC-4 互通。 但是 MAC-1 不能与 MAC-3 与 MAC-4 互通，为啥？因为 MAC-1 这块网络卡并没有与 MAC-3 及 MAC-4 使用同样的 switch/hub 相接嘛！所以，数据的流通会变成：

1. 先由 MAC-1 传送到 MAC-2 ，此时来源是 MAC-1 而目的地是 MAC-2；
2. B 计算机接收后，察看该讯框，发现目标其实是 C 计算机，而为了与 C 计算机沟通， 所以他会将讯框内的来源 MAC 改为 MAC-3 ，而目的改为 MAC-4 ，如此就可以直接传送到 C 计算机了。

也就是说，只要透过 B (就是路由器) 才将封包送到另一个网域 (IP 部分会讲) 去的时候， 那么讯框内的硬件地址就会被改变，然后才能够在同一个网域里面直接进行 frame 的流通啊！

另外，这个 MAC 讯框可以容纳多大的数据啊？？在正规的以太网络当中， 就如同上图三所标示的，一个讯框标准容量最大可达 1500Bytes， 也就是说，在整条网络上面，一个讯框最大就仅能达到 1500bytes。那如果我有 100M Bytes 的数据要传送怎办呢？那您的操作系统会主动的将该 100M bytes 的数据拆解成为多个 1500bytes 的讯框后，传送到目的地，再重新组合成为原本 100Mbytes 的档案！ 这里也就可以解释，为什么网络共享媒体一次只能有一部主机使用， 但是局域网络内的两部计算机却可以同时下载档案？这是因为『每次要发出一个讯框时， 都需要进行 CSMA/CD 的监听，而刚刚成功发出讯框的那部主机， 也需要再使用 CSMA/CD 来跟大家抢。』所以啰，谁能抢到呢？有时后因为网络太忙碌，那么 frame 与 frame 就可能会碰撞啦。

在早期 10/100 Mbps 的年代，这个 1500 bytes 的网络媒体传输数值还没有多大的影响， 但到了 gigabit 的年代，如果使用的还是 1500 bytes 时，大型的档案将会被拆解成多个 frame ， 而多个 frame 就意味着主机需要进行多次数据的拆解，网络也需要进行多次的传输。 如果可以将这个 MAC 的数据存放处加大的话，那么不就可以节省系统资源， 并且网络传递的次数也会降低，呵呵！没错～在这样的思考逻辑下，于是目前的 Gigabit Ethernet 通常都已经支持大的讯框架构，那就是 Jumbo Frame 啰～ 一般来说，只要是 Gigabit 以太网络卡都会支持 Jumbo frame 的 (请参考文末的参考文献连结)， 他的大小通常是定义在 9000 bytes 的， 不过其它的以太网络媒体可就不一定了。由于网络媒体支持 Jumbo frame 后， 他的效能是会有所改善的，所以挑选以太网络媒体时，记得查阅一下该媒体的说明喔！

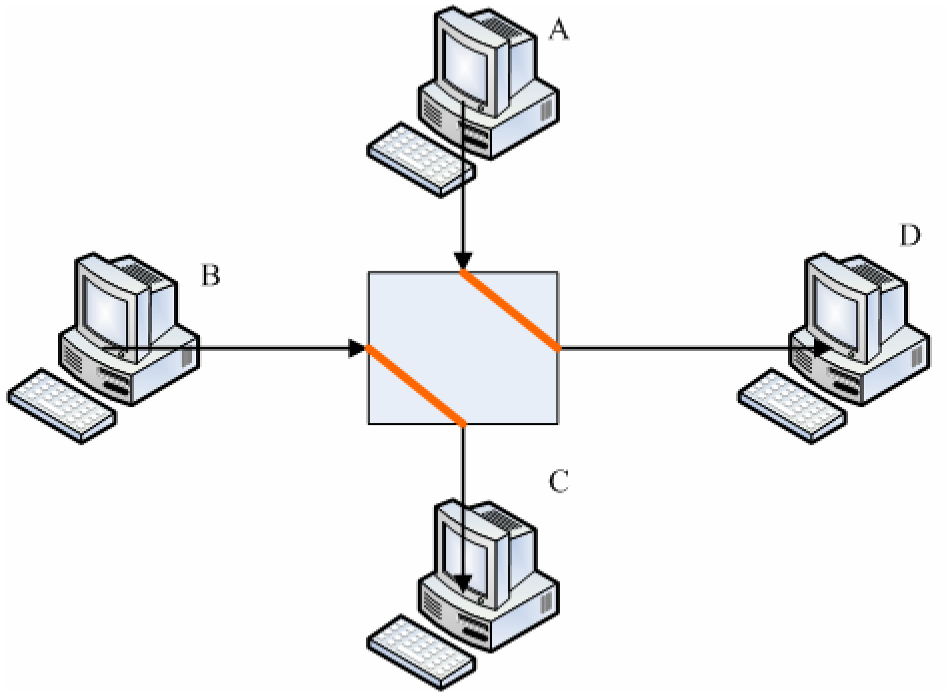
Tips:
由于网络卡卡号是跟着网络卡走的，并不会因为重灌操作系统而改变，所以防火墙软件大多也能够针对网络卡来进行抵挡的工作喔！不过抵挡网卡仅能在局域网络内进行而已，因为 MAC 不能跨 router 嘛！！



共不共享很重要，集线器还是交换器？

刚刚我们上面提到了，当一个很忙碌的网络在运作时，网络共享媒体就可能会发生碰撞的情况，这是因为 CSMA/CD 的缘故。那我们也知道在一个星形联机当中，正中央的那个设备是集线器或交换器来连接各 PC 的。那么 Hub 与 Switch 有啥不同啊？其实，那个 Hub 就是网络共享媒体，Hub 仅是将所有来自 PC 的 frame 再次送出去给所有的 PC 而已，所以他是个共享媒体。

Switch 则不然喔！Switch 内部具有微处理器以及内存，这个内存可以记录每个 switch port 与其连接的 PC 的 MAC 地址，所以，当来自 switch 两端的 PC 要互传数据时，每个 frame 将不会透过 CSMA/CD 的监听，而是透过 switch 直接将该 frame 送到目标主机上头去吶！也就是说，switch 不是个共享媒体，且 switch 的每个埠口 (port) 都具有独立的频宽喔！举例来说，10/100 的 Hub 上连结 5 部主机，那么整个 10/100Mbps 是分给这五部主机的，所以这五部主机总共只能使用 10/100Mbps 而已。那如果是 switch 呢？由于『每个 port 都具有 10/100Mbps 的频宽』，所以就看您当时的传输行为是如何啰！举例来说，如果是底下的状况时，每个联机都是 10/100 Mbps 的。



图五、Switch 的频宽简介

A 传送到 D 与 B 传送到 C 都独自拥有 10/100Mbps 的频宽，两边并不会互相影响！不过，如果是 A 与 D 都传给 C 时，由于 C port 就仅有 10/100Mbps ，等于 A 与 D 需要抢 10/100Mbps 来用的意思。总之，你就是得要记得的是，switch 已经克服了封包碰撞的问题，因为他有个 switch port 对应 MAC 的相关功能，所以 switch 并非共享媒体喔！同时需要记得的是，现在的 switch 规格很多，在选购的时候，千万记得选购可以支持全双工/半双工，以及支持 Jumbo frame 的为佳！

那什么是全双工/半双工(full-duplex, half-duplex)？ 前面谈到网络线时，我们知道八蕊的网络线实际上仅有两对被使用，一对是用在传送，另一对则是在接收。 如果两端的 PC 同时支持全双工时，那表示 Input/Output 均可达到 10/100Mbps， 亦即数据的传送与接收同时均可达到 10/100bps 的意思，总频宽则可达到 20/200Mbps 啰 (其实是有点语病的，因为 Input 可达 10/100Mbps， output 可达 10/100Mbps ， 而不是 Input 可直接达到 20/200Mbps 喔！)如果您的网络环境想要达到全双工时， 使用共享媒体的 Hub 是不可能的，因为网络线脚位的关系，无法使用共享媒体来达到全双工的！ 如果你的 switch 也支持全双工模式，那么在 switch 两端的 PC 才能达到全双工喔！

一些常见的以太网络技术

如果您常常在网络上搜寻一些硬件信息时，或者是常常跑到信息卖场去看看新鲜货时， 应该会注意到一些网络硬件，尤其是越来越普及的 switch 这玩意儿的相关硬件信息吧！ 而且，我们知道网络线因为接头的关系而有并行线与跳线，这两种网络线使用的时机并不相同， 那么你是否一定需要购买特殊的线段才能够连结 PC 与 switch 呢？呵呵！不需要～ 因为现在的硬件实在太聪明了！底下的功能您应该都可以在新的硬件上面发现的！

自动协调速度机制：

我们都知道现在的以太网络卡是可以向下支持的，亦即是 Gigabit 网络卡可以与早期的 10/100Mbps 网络卡连结而不会发生问题。但是，此时的网络速度是怎样判定呢？ 早期的 switch/hub 必须要手动切换速度才行，新的 hub/switch 因为有支持 auto-negotiation 又称为 N-Way 的功能，他可自动的协调出最高的传输速度来沟通喔！如果有 Gigabit 与 10/100Mbps 在 switch 上面， 则 N-Way 会先使用最高的速度 (gigabit) 测试是否能够全部支持，如果不行的话，就降速到下一个等级亦即 100 Mbps 的速度来运作的！

Auto MDI/MDIX：

那么我们是否需要自行分辨并行线与跳线呢？不需要啦！因为 switch 若含有 auto MDI/MDIX 的功能时，会自动分辨网络线的脚位来调整联机的，所以您就不需要管你的网络线是跳线还是并行线啰！方便吧！ ^_^

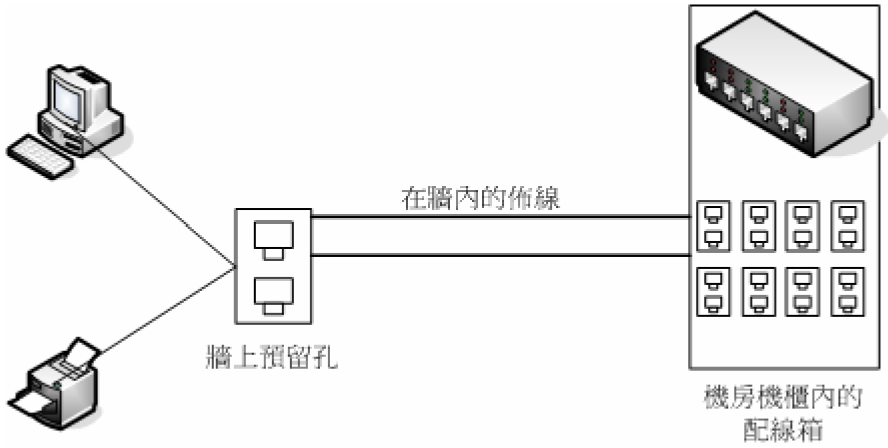
讯号衰减造成的问题

由于电子讯号是会衰减的，所以当网络线过长导致电子讯号衰减的情况严重时， 就会导致联机质量的不良了。因此，连结各个节点的网络线长度是有限制的喔！ 不过，一般来说，现今的以太网络 CAT5 等级的网络线大概都可以支持到 100 公尺的长度， 所以应该无庸担心才是呐！

但是，造成讯号衰减的情况并非仅有网络线长度而已！如果您的网络线折得太严重(例如在门边常常被门板压，导致变形) ，或者是自行压制网络线接头，但是接头部分的八蕊蕊线缠绕度不足导致电磁干扰严重， 或者是网络线放在户外风吹日晒导致脆化的情况等等，都会导致电子讯号传递的不良而造成联机质量恶劣，此时常常就会发现偶而可以联机、有时却又无法联机的问题了！因此，当您需要针对企业内部来架设整体的网络时， 注意结构化布线可是很重要的喔！

结构化布线

所谓的结构化布线指的是将各个网络的组件分别拆开，分别安装与布置到企业内部， 则未来想要提升网络硬件等级或者是移动某些网络设备时，只需要更动类似配线盘的机柜处， 以及末端的墙上预留孔与主机设备的联机就能够达到目的了。例如底下的图示：



图六、结构化布线简易图标

在墙内的布线需要很注意，因为可能一布线完成后就使用 20 年以上喔！ 那您需要注意的仅有末端墙上的预留孔以及配线端部分。 事实上，光是结构化布线所需要选择的网络媒体与网络线的等级， 还有机柜、机架，以及美化与隐藏网络线的材料等等的挑选，以及实际施工所需要注意的事项， 还有所有硬件、施工所需要注意的标准规范等等， 已经可以写满厚厚一本书，而鸟哥这里的文章旨在介绍一个中小企业内部主机数量较少的环境， 所以仅提到最简单的以一个或两个交换器 (swtich) 串接所有网络设备的小型星形联机状态而已， 如果您有需要相关硬件结构化布线的信息，可以参考风信子兄翻译的『Swtich and Fast 以太网络』一书的后半段！至于网络上的高手吗？您可以前往酷学园请教 ZMAN 大哥喔！

OSI 七层协定

目前我们的主机只要能够取得正确的 IP 与相关参数设定时，你就可以连上 Internet 了， 根本不管你的网络硬件是以太网络还是光纤网络。而且，你主机的操作系统是啥 Internet 也是不管的！这是为什么呢？因为网络的传输是有分层架构的，每个分层 (layer) 是可以独立的。 同时每个分层都有独自的标准可供依循，例如在网络媒体的硬件部分就可以参考 IEEE 的 802.3 的标准！ 如此一来，大家都可以在自己的分层当中找到相关的标准来设定自己的数据，如此网络连结就变的更容易了。 关于网络的分层我们最喜欢拿 OSI (Open system Interconnection) 七层协定来说明喔！ 事实上， OSI 七层协议只是一个参考的模型 (model)，不过，由于 OSI 所定义出来的七层协议相当良好， 所以拿来当作网络联机解释真是太棒了！底下就来说说吧！

分层	负责内容
Layer 1 实体层 Physical Layer, PHY	在这个层级当中主要定义了最基础的网络硬件标准，包括各种网络线、各种无线联机方式，各种设备规范、 以及各种接头的规则，还有传输讯号的电压等等，反正与硬件有关的标准大多都在这个层级当中定义的！
Layer 2 数据连接层 Data-Link Layer	由于传送数据的网络媒体是以是电子讯号进行传送，所以我们的数据要使用这样的讯号传送时， 就需要制订各种网络型态的讯框 (frame) 了，才能确保数据可以在不同的网络媒体进行传送的动作。 所以，在这一层当中就制订了 frame 的格式以及通过网络的方式。 包括讯框的数据格式、错误控制、流量控制、检查数据传输错误的方法等等，都在这里控制。 既然与讯框有关，当然这个层级就与前面提到的 MAC 有很强烈的相关性啰！

	但我们知道事实上目前的 Internet 使用的其实是 IP 来进行联机的啊！但硬件数据却是由讯框所传送的。 为了要将两者对应 (MAC 与 IP 的对应)，就必须要经由 Address Resolution Protocol (ARP) 这个协定来帮忙解析出对应才行！
Layer 3 网络层 Network Layer	这一层是我们最感兴趣的啰～因为我们提及的 IP (Internet Protocol) 就是在这一层定义的， 同时也定义出计算机之间的联机建立、终止与维持等，数据封包 (packet) 的传输路径选择等等， 因此这个层级当中最重要的除了 IP 之外，就是封包能否到达目的地的路由 (route) 概念了！ 此外，这一个网络层可以涵盖实体层与数据链路层，通常我们不需要设定硬件与相关 MAC 的数据， 就是因为网络层已经 (有点类似) 隐藏了底下两层，让我们只要设定好 IP 就能够上网呐！ IP 与 route 的部分我们会在下一小节加以介绍的。
Layer 4 传送层 Transport Layer	这一个分层定义了发送端与接收端的联机技术(如 TCP 技术)， 同时包括该技术的封包格式，数据封包的传送、流程的控制、传输过程的侦测检查与复原重新传送等等， 以确保各个资料封包可以正确无误的到达目的端。
Layer 5 会谈层 Session Layer	在这个层级当中主要定义了两个地址之间的联机信道之连接与挂断，此外，亦可建立应用程序之对谈、 提供其它加强型服务如网络管理、签到签退、对谈之控制等等。如果说传送层是在判断资料封包是否可以正确的到达目标， 那么会谈层则是在确定网络服务建立联机的确认，例如三向交握。这部分我们会在底下的 TCP 技术当中做个说明。
Layer 6 表现层 Presentation Layer	我们在应用程序上面所制作出来的数据格式不一定符合网络传输的标准编码格式的！ 所以，在这个层级当中，主要的动作就是：将来自本地端应用程序的数据格式转换(或者是重新编码)成为网络的标准格式， 然后再交给底下传送层等的协议来进行处理。所以，在这个层级上面主要定义的是网络服务(或程序)之间的数据格式的转换， 包括数据的加解密也是在这个分层上面处理。
Layer 7 应用层 Application Layer	完全与程序有关的啰，包括定义出档案的读取、复制、开启、关闭等等，常见的程序包括有浏览器、数据库处理系统与电子邮件系统等等。

事实上，在上述的七层协议当中，前两层(实体与数据连接层)主要就是由一些硬件标准所规范出来的， 像我们前一小节提到的以太网络之 MAC 讯框相关的格式，以及一些类似以太网络线接头规范、 CSMA/CD 的技术等等，都是在前两层进行规范的。

至于网络层与传送层则与 TCP/IP 有关。我们知道目前的 Internet 相关的 IP 与 TCP 封包格式是由 Internet Network Information Center (INTERNIC) 所统一整理与维护的，至于 TCP/IP 的标准则主要以 Request For Comment (RFC) 技术报告的形式公开。 而会谈、表现与应用层则主要与操作系统及应用程序有关了。

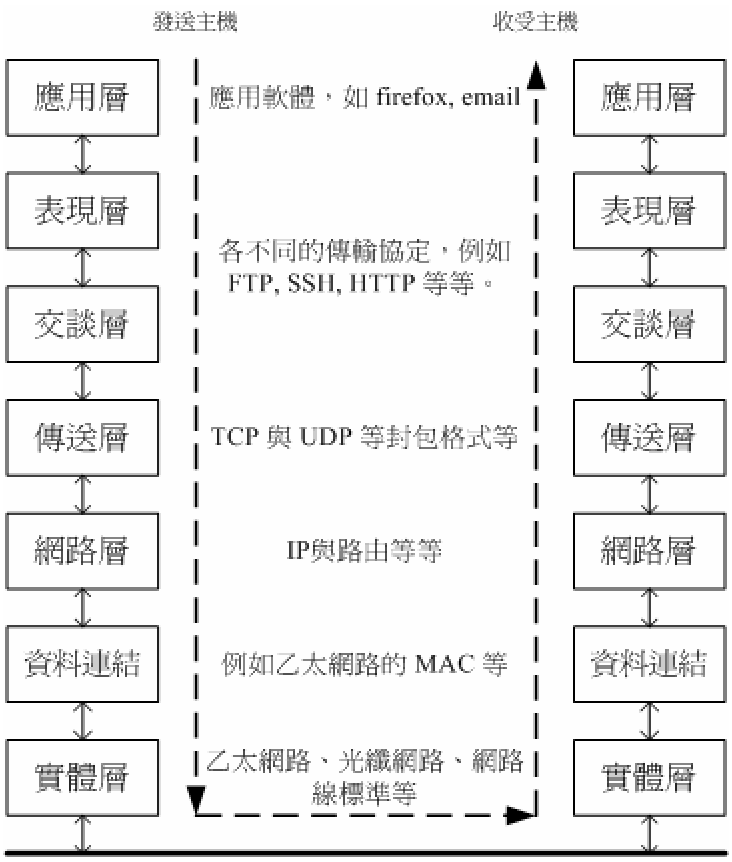
那么这七层到底是如何运作的呢？我们以常见的 WWW 浏览器来进行说明好了。 假设你想要由奇摩雅虎

(tw.yahoo.com)下载一个大于 10Mbytes 的档案， 那么你必须要由你的主机打开浏览器，并且输入相关的网址列后才能开始下载，对吧！ 不过，我们知道由于网络媒体的关系，标准以太网络的硬件最大仅能支持 1500 bytes 的讯框大小， 而我要去奇摩雅虎时，必须要知道奇摩雅虎那部主机的 IP 才行，而我们的浏览器使用的是 TCP 的封包格式。这样一层一层下来，你可以将各个分层想成是一个一个的大袋子， 而且每个袋子都必须要包含在下一个袋子内，例如 IP 的袋子必须要装在 MAC 讯框的袋子内。

- 所以，由于最后被传送的袋子(实体层)的限制，我必须要将 10Mbytes 的档案先切成数个小包，然后将这些小包给他包到 TCP 的袋子内，这个袋子记载了我的数据内容。
- 然后这个袋子还要装到 IP 这个袋子内，IP 这个袋子会记录我的住址以及要传送到的目的地的住址，
- 最后再将这个 IP 的袋子装到 MAC 的讯框袋子内， 这个袋子就记录了可以在同一区域内传递袋子的网卡卡号了。

之后这个 MAC 的大袋子就会被带到下个传递点去，当然啦，MAC 这个袋子的传递是需要符合 CSMA/CD 以及以太网络的相关定义的喔！ 当传到目的地后，对方会一个袋子一个袋子的解开，最后拿到他们的数据。不过，由于我的 10Mbytes 已经被分成多个小袋子了，所以每个小袋子内需要有序号，这样当所有的小袋子都到达目的地后， 对方才可以依照这些序号将所有小袋子内的数据给他整合成为原来的数据啊！

所以啊，这些分层可以使用底下的图示来看：



图七、OSI 七层协议的相关性

例题一：请找出您 Linux 主机上面的网络卡硬件地址(Hardware Address, 或 MAC)，如果已经

连上网络的话，请找出您局域网络内其它计算机的网络卡卡号。

答：

在 Linux 底下网络卡的装置代号一般是 eth0 ，所以想要了解您的网络卡卡号时，可以使用：『 ifconfig eth0 』这个指令来查阅，在出现的数据中第一行最右边搜寻 HWaddr 的后面接的那串咚咚，就是你的卡号。至于其它的卡号与 IP 的对应方面，直接输入『 arp -n 』应该就可以查阅的到相关的对应表啰！更多说明请先使用 man 来查询，后续章节我们也会继续加以介绍的。

IP 与 MAC

我们现在知道要有网络的话，必须要有网络相关的硬件，而目前最常见的网络硬件接口为以太网络， 包括网络线、网络卡、Hub/Switch 等等。而以太网络上面的传输使用网络卡卡号为基准的 MAC 讯框， 配合 CSMA/CD 的监听技术来传送讯框，这是硬件部分。那么在软件部分，我们知道 Internet 其实就是 TCP/IP 这个通讯协议的通称，Internet 是由 INTERNIC 所统一管理的， 但其实他仅是负责分配 Internet 上面的 IP 以及提供相关的 TCP/IP 技术文件而已， 另外，在 TCP/IP 上面还有很多的应用程序，包括 FTP, HTTP, EMAIL 等等的技术！ 底下我们就先来谈一谈最底层的 MAC 与 IP 吧！

传输单位与 MAC

想一想，如果没有电的话，我们的网络是否能够通行无阻？当然不行！ 因为网络其实就是电子讯号的传送啊！如果没有电，当然就没有办法传送讯息了。而电子讯号只有 0 跟 1 啊，所以在网络单位的计算上，一般通常是以二进制的 bit 为单位的。那么这个 bit 与我们通常用来计算档案大小的单位 bytes 有什么关连性？其实：

1byte = 8bits

所以啦，一般来说，我们看到的网络提供者 (Internet Services Provider, ISP) 所宣称他们的 ADSL 传输速度可以达到 下行/上行 2Mbps/128Kbps (Kbits per second) 时，那个 Kb 指的可不是 bytes 而是 bits 喔!所以 2M/128K 在实际的档案大小传输速度上面,最大理论的传输为 256KBps/16 KBps(KBytes per second)，所以正常下载的速度约在每秒 100~200 KBytes 之间呐！ 同样的道理，在网络卡或者是一些网络媒体的广告上面，他们都会宣称自己的产品可以自动辨识传输速度为 10/100 Mbps (Mega-bits per second)，呵呵！该数值还是得再除以 8 才是我们一般常用的档案容量计算的单位 bytes 喔！这样可以了解传输单位的意义了吗？！

那么 MAC 呢？MAC 是 Media Access Control (媒体存取控制)，我们在前小节的 图三已经介绍过该讯框的数据格式， 且 MAC 常用来做为硬件地址 (Hardware address) 的代称。我们自己主机上面的 MAC 很好解决， 假设您的网络卡仅有一张，则 Linux 系统内网卡的代号预设由 eth0 开始编号， 因此当你输入 ifconfig eth0 这个指令时，就会出现如下的讯息了：

```
在 Linux 环境下
[root@linux ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:01:03:43:E5:34
          inet addr:192.168.1.100   Bcast:192.168.1.255   Mask:255.255.255.0
          inet6 addr: fe80::201:3ff:fe43:e534/64 Scope:Link
```

```
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
.....

在 Windows 环境下
C:\Documents and Settings\admin..> ipconfig /all
....
        Physical Address. . . . . . . . . : 00-01-03-43-E5-34
....
```

事实上，这个 MAC 几乎都是焊死在网络卡上面的，所以不能够被修改。 不过，近来有些笔记型计算机上面的网络卡可以透过软件来进行 MAC 的修改啰～ 好，在我们的主机上网络卡可以透过 ifconfig 来查询到 MAC ，但我们知道讯框是在两张网络卡之间传讯的， 那我如何知道其它主机的网络卡卡号呢？此时就得要透过 ARP (Address Resolution Protocol) 的帮忙了。 由于 TCP/IP 的通讯协议内大多仅需要了解 IP 即可，但讯框却是透过 MAC 来传递， 因此 IP 与 MAC 就得要透过一个解析的功能啰！那就是 ARP 啦！

当我们的主机想要找出目标 IP 时，就会对整个局域网络进行广播封包(broadcast)的传送， 这个广播封包可以对所有局域网络内的计算机要求回报他的 IP 与 MAC ， 当目标 IP 看到这个广播封包时，就会响应您主机相关的 MAC 信息，如果非目标主机接到这个封包， 就会主动的忽略！如此一来，你就可以取得目标主机的 MAC 啰！ 而这个目标主机的 MAC 就会被记录到你的主机内的 ARP table (ARP table 在内存中)， 不过还是要再次的提醒， MAC 是不能跨路由的，请参考图四的相关说明喔！ 如果想要查阅你的 ARP 记录，可以使用 arp 这个指令即可。

```
[root@linux ~]# arp -[nd] hostname
[root@linux ~]# arp -s hostname(IP) Hardware_address
参数：
-n ：将主机名称以 IP 的型态显示
-d ：将 hostname 的 hardware_address 由 ARP table 当中删除掉
-s ：设定某个 IP 或 hostname 的 MAC 到 ARP table 当中
范例一：
[root@linux ~]# arp -n
Address            HWtype  HWaddress           Flags Mask    Iface
192.168.1.100      ether   00:01:03:01:02:03      C          eth0
192.168.1.240      ether   00:01:03:01:DE:0A      C          eth0
192.168.1.254      ether   00:01:03:55:74:AB      C          eth0
范例二：
[root@linux ~]# arp -s 192.168.1.100  01:00:2D:23:A1:0E
# 这个指令的目的在建立静态 ARP
```

如同上面的表格，我主机上面的 ARP 记录着这么多的 IP 与 MAC 的对应， 这个 ARP 的好处可多了！由于有记录 MAC 与 IP 的对应，因此当下回我的数据又传送到同一部主机时， 我的主机会主动的传送到下一个 MAC 去，而不需要再次透过 broadcast 来查询 MAC， 所以省去了很多网络延迟的时间喔！此外，特别注意的是， ARP table 是动态的信息，他会随时随着您的网域里面计算机的 IP 更动而变化，所以，即使您常常更动您的计算机 IP，不要担心，因为 ARP table 会自动的重新对应 IP 与 MAC 的表格内容！但如果你有特殊需求的话， 也可以利用『 arp -s 』这个参数来定义静态的 ARP 对应喔！

IP 的组成

好了，接下来可以开始介绍那个可爱的 TCP/IP 里头的 IP 啦！这个 IP 是 Internet Protocol 的缩写，他的功能有点像是『门牌号码』，主要是在网络层(Layer 3)的功能， 那么这个 ? 有哪些重要的地方需要了解的呢？底下我们就来谈一谈吧！

IP 是一种数据封包的格式，这个 IP 数据封包最大可以到达 65535 bytes ， 然而就如同图三以太网络讯框的数据所示， 由于标准以太网络讯框可包含的数据最大仅达 1500 bytes ， 并且依照不同的网络媒体而有不同的 MAC 讯框大小，我们前面也谈到的 OSI 七层协议当中， 由于 IP 封包必须要放到 MAC 讯框当中，因此 IP 封包在 Internet 上面应该是不可能达到 65535 bytes 这个值的(因为必须小于 MAC 讯框所能容许的最大值)。另外， 由于网络联机过程当中封包所经过的网络媒体各不相同，因此 MAC 讯框大小当然也不同， 而为了让 IP 封包可以适用在所有的网络媒体讯框当中，因此， IP 封包是可以被『重组的』！

我们知道 MAC 讯框表头 (将他想成是一个信封袋外面的记录数据) 当中最重要的就是网络卡卡号 (hardware address) 这个咚咚！(参考图三)，那么 IP 表头最重要的是什么呢？呵呵！那就是 IP 地址 (address) 了！目前我们在 Internet 上面使用的 IP 协议是第四版，通称为 IPv4 ，这个版本的 IP 地址主要是由 32 bits 的数据所组成的一组数据，也就是 32 个 0 跟 1 所组成的数据数据，因为只有零跟一，所以 IP 的组成当然就是计算机认识的二进制的表示方式了。

不过，因为人类对于二进制实在是不怎么熟悉，所以为了顺应人们对于十进制的依赖性，因此，就将 32 bits 的 IP 分成四小段，每段含有 8 个 bits ，将 8 个 bits 计算成为十进制，并且每一段中间以小数点隔开，那就成了目前大家所熟悉的 IP 的书写模样了。如下所示：

```
IP 的表示式：
00000000.00000000.00000000.00000000   ==> 0.0.0.0
11111111.11111111.11111111.11111111   ==> 255.255.255.255
```

所以 IP 最小可以由 0.0.0.0 一直到 255.255.255.255 哩！事实上， IP 的组成当中，除了以 32 bits 的组成方式来说明外，还具有所谓的『网域』的概念存在。底下就来谈一谈什么是网域吧！

网域的概念与 IP 的分级

事实上在 IP 的 32 bits 资料中，主要分为 HOST_ID 与 Net_ID 两部份，我们先以 192.168.0.0 ~ 192.168.0.255 这个 C Class 的网域当作例子来说明好了：

```
192.168.0.0~192.168.0.255 这个 C Class 的说明：
11000000.10101000.00000000.00000000
11000000.10101000.00000000.11111111
|----------Net_ID---------|-host--|
```

在 C Class 的范例当中，前面三组数字 (192.168.0) 称为网域号码 (Net_ID) ，最后面一组数字则称为主机号码 (Host_ID)。同一个网域当中的定义是『在同一个物理网段内，主机的 IP 具有相同的 Net_ID ，并且具有独特的 Host_ID』，那么这些 IP 群就是同一个网域内的 IP 网段啦！

Tips:
什么是物理网段呢？当所有的主机都是使用同一个网络媒体串在一起， 这个时候这些主机在实体装置上面其实是联机在一起的，那么就可以称为这些主机在同一个物理网段内了！ 同时并请注意，同一个物理网段之内，可以依据不同的 IP 的设定，而设定成多个『IP 网段』喔！



上面例子当中的 192.168.0.1, 192.168.0.2, ..., 192.168.0.255 这些 IP 就是同一个网域内的 IP 群 (同一个网域也称为同一个网段！)，请注意，同一个 Net_ID 内，不能具有相同的 Host_ID ，否则就会发生 IP 冲突，可能会造成两部主机都没有办法使用网络的问题！ 那么同一个网域该怎么设定，与将 IP 设定在同一个网域之内有什么好处呢？

- 在同一个网段内，Net_ID 是不变的，而 Host_ID 则是不可重复，此外， Host_ID 在二进制的表示法当中，不可同时为 0 也不可同时为 1 ，例如上面的例子当中， 192.168.0.0 (Host_ID 全部为 0)以及 192.168.0.255 (Host_ID 全部为 1) 不可用来作为网段内主机的 IP 设定，也就是说，这个网段内可用来设定主机的 IP 是由 192.168.0.1 到 192.168.0.254;

- 在同一个网域之内，每一部主机都可以透过 MAC 讯框的格式传递资料， 并透过 ARP 协议与广播封包 (broadcast) 取得 MAC 与 IP 的对应后，直接利用 MAC 讯框传递数据。

- 在同一个物理网段之内，如果两部主机设定成不同的 IP 网段，则两部主机无法直接以 MAC 讯框格式进行数据的传递，因为广播封包无法查询到 MAC 与 IP 的对应。

- 当 Host_ID 所占用的位越大，亦即 Host_ID 数量越多时，表示同一个网域内可用以设定主机的 IP 数量越多。

所以说，贵单位公司内的计算机群，或者是您宿舍或家里面的所有计算机，当然都设定在同一个网域内是最方便的， 因为如此一来每一部计算机都可以直接透过 MAC 来进行数据的交流，而不必经由 Router (路由器) 来进行封包的转递呢！(Router 这部份在后续才会提及！)。

IP 的分级

好了，现在我们知道 Net_ID 越大时，表示 Host_ID 越少，亦即网域内可以分配的 IP 数量就越少了！ 咦！这表示 Net_ID 是有分级的喔！是啊！没错～刚刚上面那个 192.168.0.0~192.168.0.255 称为 Class C，那还有哪些等级啊？目前 Internet 将 IP 简单的分类成为三种常见的等级，亦即所谓的 A, B, C class，他们代表的意义如下：

```
以二进制说明 Network 第一个数字的定义：
A Class : 0xxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx  ==> NetI_D 的开头是 0
          |--net--|---------host------------|
B Class : 10xxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx  ==> NetI_D 的开头是 10
          |------net-------|------host------|
C Class : 110xxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx  ==> NetI_D 的开头是 110
          |-----------net-----------|-host--|

三种分级在十进制的表示：
A Class :   0.xx.xx.xx ~ 126.xx.xx.xx
```

```
B Class : 128.xx.xx.xx ~ 191.xx.xx.xx
C Class : 192.xx.xx.xx ~ 223.xx.xx.xx
```

在上表中，可能您会觉得很奇怪，咦！那个 127.xx.xx.xx 怎么不见了？！他应该也是 A Class 的一段吧？！没错，是不见了， 因为这个网段被拿去给操作系统做为内部循环网络 (loopback) 之用了！在各个操作系统当中，不管该主机的硬件有没有网络卡， 为了让作业确认自己的网络没有问题，所以将 127.xx.xx.xx 这个 A Class 的网段拿到操作系统当中，来做为内部的回路测试！所以啦，这个 127.0.0.1 就不可以用来做为其它网络卡的网络网域之设定喔。

Netmask 的用途与子网络的切分

在上一小节当中提到的 A, B, C 三个层级的网域是由 IP 协定预设分配的，在这样的层级当中，我们可以发现 A Class 可以用于设定计算机主机的 IP 数量 (Host) 真的是很多，在同一个 A Class 的网域内，主机的数量可以达到『 256 X 256 X 256 - 2(Host_ID 全为 0 或 1) = 16777214』，不过， 这样的设定情况对于一般网络的效能却是不太好的！为什么呢？

让我们回到前面以太网络的 MAC 运作模式那个小节，我们知道在共享媒体上面， 每当任何一部主机想要使用该网络媒体时，就得要利用 CSMA/CD 的方式去进行网络监听的工作，此时对于这么大的一个网络架构来说，每部主机要发出 MAC 讯框前要进行的这个 CSMA/CD 实在会造成系统上面很严重的停顿问题啊！ 因为封包碰撞 (collision) 以及在进行 MAC 与 IP 对应的广播 (broadcast) 时， 要响应的主机数量也真是太多了点吧！ 如此一来，整个网络的效能将会变的很糟糕！所以，一般来说，我们最多都仅设定 C Class 做为整个局域网络的架构，其实就连 C Class 也都太大了！不过不打紧，只要记得一个网域内不要超过 30 部以上的主机数量，那样网络的效能就会比较好一点～

其实，除了 C Class 之外，我们还是可以继续将网络切的更细的！上个小节我们提到 IP 这个 32 bits 的数值中分为 Net_ID 与 Host_ID，其中 C Class 的 Net_ID 占了 24 bits ，而其实我们还可以将这样的网域切的更细，就是让第一个 Host_ID 被拿来作为 Net_ID ，所以，整个 Net_ID 就有 25 bits ，至于 Host_ID 则减少为 7 bits 。在这样的情况下，原来的一个 C Class 的网域就可以被切分为两个子网域，而每个子网域就有『 256/2 - 2 = 126 』个可用的 IP 了！这样一来，在这个网域当中的主机在进行逻辑广播时，响应的主机数量就少了一半， 当然对于网络的效能多多少少有点好处的啦！

好了，知道了子网络切分的大致情况后，现在要谈的是，那么到底是什么参数来达成子网络的切分呢？呵呵！那就是 Netmask (子网掩码) 的用途啦！这个 Netmask 是用来定义出网域的最重要的一个参数了！不过他也最难理解了～ @_@。为了帮助大家比较容易记忆住 Netmask 的设定依据，底下我们介绍一个比较容易记忆的方法。同样以 192.168.0.0~192.168.0.255 这个网域为范例好了，如下所示，这个 IP 网段可以分为 Net_ID 与 Host_ID，既然 Net_ID 是不可变的，那就假设他所占据的 bits 已经被用光了 (全部为 1)，而 Host_ID 是可变的，就将他想成是保留着 (全部为 0)，所以， Netmask 的表示就成为：

```
192.168.0.0~192.168.0.255 这个 C Class 的 Netmask 说明
11000000.10101000.00000000.00000000
11000000.10101000.00000000.11111111
|----------Net_ID---------|-host--|
11111111.11111111.11111111.00000000  <== Netmask 二进制
  255   .  255   .  255   .   0      <== Netmask 十进制
```

将他转成十进制的话，就成为『255.255.255.0』啦！ 这样记忆简单多了吧！照这样的记忆方法，那么 A, B, C Class 的 Netmask 表示就成为这样：

```
Class A, B, C 三个等级的 Netmask 表示方式：
A Class : 11111111.00000000.00000000.00000000 ==> 255.  0.  0.  0
B Class : 11111111.11111111.00000000.00000000 ==> 255.255.  0.  0
C Class : 11111111.11111111.11111111.00000000 ==> 255.255.255.  0
```

所以说， 192.168.0.0~192.168.0.255 这个 C Class 的网域中，他的 Netmask 就是 255.255.255.0 了！再来，我们刚刚提到了当 Host_ID 全部为 0 以及全部为 1 的时后该 IP 是不可以使用的，因为 Host_ID 全部为 0 的时后，表示 IP 是该网段的 Network ，至于全部为 1 的时后就表示该网段最后一个 IP ，也称为 Broadcast ，所以说，在 192.168.0.0 ~ 192.168.0.255 这个 IP 网段里面的相关网络参数就有：

```
Netmask:   255.255.255.0   <==网域定义中，最重要的参数
Network:   192.168.0.0     <==第一个 IP
Broadcast: 192.168.0.255   <==最后一个 IP
可用以设定成为主机的 IP 数：
192.168.0.1 ~ 192.168.0.254
```

一般来说，如果我们知道了 Network 以及 Netmask 之后，就可以定义出该网域的所有 IP 了！因为由 Netmask 就可以推算出来 Broadcast 的 IP 啊！因此，我们常常会以 Network 以及 Netmask 来表示一个网域，例如这样的写法：

```
Network/Netmask
192.168.0.0/255.255.255.0
192.168.0.0/24      <==因为 Net_ID 共有 24 个 bits
```

另外，既然 Netmask 里面的 Net_ID 都是 1 ，那么 C Class 共有 24 bits 的 Net_ID ，所以啦，就有类似上面 192.168.0.0/24 这样的写法啰！这就是一般网域的表示方法。好了，刚刚提到 C Class 还可以继续进行子网域 (Subnet) 的切分啊，以 192.168.0.0/24 这个情况为例，他要如何再细分为两个子网域呢？我们已经知道 Host_ID 可以拿来当作 Net_ID，那么 Net_ID 使用了 25 bits 时，就会如下所示：

```
原本的 C Class 的 Net_ID 与 Host_ID 的分别
11000000.10101000.00000000.00000000      Network:   192.168.0.0
11000000.10101000.00000000.11111111      Broadcast: 192.168.0.255
|----------Net_ID---------|-host--|

切成两个子网络之后的 Net_ID 与 Host_ID 为何？
11000000.10101000.00000000.0 0000000  多了一个 Net_ID 了, 为 0
11000000.10101000.00000000.1 0000000  多了一个 Net_ID 了, 为 1
|----------Net_ID-----------|-host--|

第一个子网络
Network:   11000000.10101000.00000000.0 0000000   192.168.0.0
Broadcast: 11000000.10101000.00000000.0 1111111   192.168.0.127
           |----------Net_ID-----------|-host-|
Netmask:   11111111.11111111.11111111.1 0000000   255.255.255.128
所有 IP 与网域表示式：
```

```
192.168.0.0/25 或 192.168.0.0/255.255.255.128

第二个子网络
Network:   11000000.10101000.00000000.1 0000000   192.168.0.128
Broadcast: 11000000.10101000.00000000.1 1111111   192.168.0.255
           |----------Net_ID-----------|-host-|
Netmask:   11111111.11111111.11111111.1 0000000   255.255.255.128
所有 IP 与网域表示式：
192.168.0.128/25 或 192.168.0.128/255.255.255.128
```

所以说，当再细分下去时，就会得到两个子网域，而两个子网域还可以再细分下去喔 (Net_ID 用掉 26 bits ....)。呵呵！如果您真的能够理解 IP, Network, Broadcast, Netmask 的话，恭喜您，未来的服务器学习之路已经顺畅了一半啦！ ^_^

例题二：请试着计算出 172.16.0.0/23 这个网域的 Netmask, Network, Broadcast 等参数

答：

由于 172.16.xxx.xxx 是在 Class B 的等级当中，亦即 172.16.0.0/16 才对。不过题目中询问的是 172.16.0.0/23，等于是向 Host_ID 借了 7 个 bits 用在 Net_ID 当中。所以整个 IP 的地址会变成这样：

```
预设：        172 .  16   .0000000 0.00000000
          |----Net_ID--------------|--Host---|
Network:      172 .  16   .0000000 0.00000000   172.16.0.0
Broadcast:    172 .  16   .0000000 1.11111111   172.16.1.255
Netmask:   11111111.11111111.1111111 0.00000000   255.255.254.0
```

鸟哥在这里有偷懒，因为这个 IP 段的前 16 个 bits 不会被改变，所以并没有计算成二进制(172.16)，真是不好意思啊～至于粗体部分则是代表 host_ID 啊！

IP 的种类与取得方式

接下来要跟大家谈一谈也是很容易造成大家困扰的一个部分，那就是 IP 的种类！很多朋友常常听到什么『真实 IP, 实体 IP, 虚拟 IP, 假的 IP....』烦都烦死了～其实不要太紧张啦！实际上，在 IPv4 里面就只有两种 IP 的类别，分别是：

- Public IP : 公共 IP ，经由 INTERNIC 所统一规划的 IP， 有这种 IP 才可以连上 Internet ；
- Privat IP : 私有 IP 或保留 IP，不能直接连上 Internet 的 IP ， 主要用于局域网络内的主机联机规划。

早在 IPv4 规划的时候就担心 IP 会有不足的情况，而且为了应付某些私有网络的网络设定，于是就有了私有 IP (Private IP) 的产生了。私有 IP 也分别在 A, B, C 三个 Class 当中各保留一段作为私有 IP 网段，那就是：

- A Class：10.0.0.0   - 10.255.255.255

- B Class：172.16.0.0 - 172.31.255.255
- C Class：192.168.0.0 - 192.168.255.255

由于这三个 Class 的 IP 是预留使用的，所以并不能直接作为 Internet 上面的连接之用，不然的话，到处都有相同的 IP 啰！那怎么行！网络岂不混乱？所以啰，这三个 IP 网段就只做为内部私有网域的 IP 沟通之用，也就是说，他有底下的几个限制：

- 私有地址的路由信息不能对外散播 （就是内部网络咯）；
- 使用私有地址作为来源或目的地址的封包，不能透过 Internet 来转送 （呵呵！当然啰！不然网络会混乱）；
- 关于私有地址的参考纪录(如 DNS)，只能限于内部网络使用（一样的原理啦！）

这个私有 IP 有什么好处呢？呵呵！由于他的私有路由不能对外直接提供信息，所以呢， 你的内部网络将不会直接被 Internet 上面的 Cracker 所攻击！但是，你也就无法以私有 IP 来『直接上网』啰！所以相当适合一些尚未具有 Public IP 的企业内部用来规划其网络之设定之用！否则当你随便指定一些可能是 Public IP 的网段来规划你企业内部的网络设定时，万一哪一天真的连上 Internet 了，那么启不是可能会造成跟 Internet 上面的 Public IP 相同了吗？ 这可不是闹着玩的，要将你原先规划的 IP 网段整个重新调整过呢！哈哈！累死了！

那么万一你又要将这些私有 IP 送上 Internet 呢？呵呵！这个简单，设定一个简单的防火墙加上 NAT (Network Address Transfer) 主机设定，你就可以透过 IP 伪装(不要急，这个在后面也会提到！)来使你的私有 IP 的计算机也可以连上 Internet 啰(虽然不是真的直接，但是很像『直接上网』呢！)

特殊的 loopback IP 网段
好了，那么除了这个预留的 IP 网段的问题之外，还有没有什么其它的怪东西呢？呵呵！ 当然是有啦！不然鸟哥干嘛花时间来唬 XX 呢！？没错，还有一个奇怪的 A Class 的网域，那就是 lo 这个奇怪的网域啦 (注意：是小写的 o 而不是零喔！)这个 lo 的网络是当初被用来作为测试操作系统内部循环所用的一个网域， 同时也能够提供给系统内部原本就需要使用网络接口的服务(daemon)所使用。

简单的说，如果你没有安装网络卡在的机器上面， 但是你又希望可以测试一下在你的机器上面设定的资料到底可不可以被执行，这个时候怎么办， 嘿嘿！就是利用这个所谓的内部循环网络啦！这个网段在 127.0.0.0/8 这个 A Class ，而且预设的主机 (localhost) 的 IP 是 127.0.0.1 呦！所以啰，当你启动了你的 WWW 服务器，然后在你的主机的 X-Window 上面执行 http://localhost 就可以直接看到你的主页啰！而且不需要安装网络卡呢！测试很方便吧！

此外，你的内部使用的 mail 怎么运送邮件呢？例如你的主机系统如何 mail 给 root 这个人呢？嘿嘿！也就是使用这一个内部循环啦！当要测试你的 TCP/IP 封包与状态是否正常时，可以使用这个呦！(所以哪一天有人问你嘿！你的主机上面没有网络卡， 那么你可以测试你的 WWW 服务器设定是否正确吗？这个时候可得回答：当然可以啰！使用 127.0.0.1 这个 Address 呀！ ^_^...)

IP 的取得方式
谈完了 IP 的种类与等级还有相关的子网域概念后，接下来我们得来了解一下，那么主机的 IP 是如何设定的呢？ 基本上，主机的 IP 与相关网域的设定方式主要有：

- 固定制手动设定(static)： 我们可以取得固定的 Public IP ，取得的管道可以是学术网络、或者是向 ISP 注册固定的 Public IP。不过，在使用固定的 Public IP 时，你必须要手动的在你的操作系统设定好网络参数；

- 浮动式拨接(ADSL)：除了上述的方法之外，传统的以调制解调器拨接，以及目前很流行的 ADSL 拨接，都是另一个取得 Public IP 的方法，只不过由于这种拨接的方法中，取得的 IP 是由 ISP 随机提供的，因此每次拨接到 ISP 后取得的 IP 可能都不是固定的，所以也有人称这种取得 IP 的方式为浮动式；

- 缆线(Cable modem)： 利用单向或者是双向 Cable 也可以向 ISP 注册取得 Public IP；

请记得，IP 就只有 Public 与 Private IP 这两种，而由于取得 IP 的方法不同，可能又有人喜欢宣称浮动式、固定制、动态等等的 IP ， 这很容易造成刚刚接触网络的朋友们的困扰！所以这里仅告诉大家记得『Public 与 Private IP』而已！你只要记得他就对了！其它的，以后自然就会理解的啦！^_^

IP 封包的表头

现在我们知道 IP 这个数据封包 (packet) 是需要放置在 MAC 讯框里面的，所以当然不能比 MAC 所能容许的最大数据量还大！但是 IP 封包其实可以到 65535 bytes 那么大的呐！ 那么 IP 封包除了数据之外，他的表头数据 (head) 是长怎样呢？ 在图三的 MAC 讯框表头里面最重要的莫过于那个网络卡硬件地址，那么在 IP 表头里面当然就以来源与目标的 IP 地址为最重要啰！ 除此之外， IP 表头里面还含有哪些重要数据呢？如底下所示：(下图第一行为每个字段的 bit 数)

4 bits	4 bits	8 bits	3 bits	13 bits
Version	IHL	Type of Service	Total Length	
Identification			Flags	Fragmentation Offset
Time To Live		Protocol	Header Checksum	
Source Address				
Destination Address				
Options				Padding
Data				

图八、IP 封包的表头资料

在上面的图示中有个地方要注意，那就是『每一行所占用的位数为 32 bits』， 也就是说， IP 封包的表头数据是 32 bits 的倍数喔！那各个表头的内容分别介绍如下：

- Version(版本)
  宣告这个 IP 封包的版本，例如目前惯用的还是 IPv4 这个版本，在这里宣告的。

- IHL(Internet Header Length, IP 表头的长度)
  告知这个 IP 封包的表头长度，单位为字节(bytes)。 此 IHL 长度的范围为 5~15。

- Type of Service(服务类型)
  这个项目的内容为『PPPDTRUU』，表示这个 IP 封包的服务类型，主要分为：
  PPP：表示此 IP 封包的优先度； D：若为 0 表示一般延迟(delay)，若为 1 表示为低延迟；
  T：若为 0 表示为一般传输量 (throughput)，若为 1 表示为高传输量；
  R：若为 0 表示为一般可靠度(reliability)，若为 1 表示高可靠度。
  UU：保留尚未被使用。
  我们前面谈到 gigabit 以太网络时曾提到 Jumbo frame 对吧！可以提高 MTU， 由于 gigabit 以太网络的种种相关规格可以让这个 IP 封包加速且降低延迟， 某些特殊的标志就是在这里说明的。

- Total Length(总长度)
  指这个 IP 封包的总容量，包括表头与内容 (Data) 部分。最大可达 65535 bytes。

- Identification(辨别码)
  我们前面提到 IP 袋子必须要放在 MAC 袋子当中。不过，如果 IP 袋子太大的话，就得先要将 IP 再重组成较小的袋子然后再放到 MAC 当中。而当 IP 被重组时，每个来自同一笔数据的小 IP 就得要有个识别码以告知接收端这些小 IP 其实是来自同一个封包才行。 也就是说，假如 IP 封包其实是 65536 那么大 (前一个 Total Length 有规定)， 那么这个 IP 就得要再被分成更小的 IP 分段后才能塞进 MAC 讯框中。那么每个小 IP 分段是否来自同一个 IP 资料，呵呵！这里就是那个识别码啦！

- Flags(特殊旗标)
  这个地方的内容为『0DM』，其意义为：
  D：若为 0 表示可以分段，若为 1 表示不可分段
  M：若为 0 表示此 IP 为最后分段，若为 1 表示非最后分段。

- Fragment Offset(分段偏移)
  表示目前这个 IP 分段在原始的 IP 封包中所占的位置。 就有点像是序号啦，有这个序号才能将所有的小 IP 分段组合成为原本的 IP 封包大小嘛！ 透过 Total Length, Identification, Flags 以及这个 Fragment Offset 就能够将小 IP 分段在收受端组合起来啰！

- Time To Live(TTL, 存活时间)
  表示这个 IP 封包的存活时间，范围为 0-255。当这个 IP 封包通过一个路由器时， TTL 就会减一，当 TTL 为 0 时，这个封包将会被直接丢弃。说实在的，要让 IP 封包通过 255 个路由器，还挺难的～ ^_^

- Protocol Number(协定代码)
  由于网络上面的封包协议太多了，每个协定都是装在 IP 当中的， 所以 IP 当然就得在表头上面告知收受端，这个 IP 内含有的数据是什么协议才行。 一般常见的网络协议如下所示：

IP 内的号码	协议名称(全名)
1	ICMP (Internet Control Message Protocol)
2	IGMP (Internet Group Management Protocol)
3	GGP (Gateway-to-Gateway Protocol)

4	IP (IP in IP encapsulation)
6	TCP (Transmission Control Protocol)
8	EGP (Exterior Gateway Protocol)
17	UDP (User Datagram Protocol)

- 当然啦，我们比较常见到的还是那个 TCP, UDP, ICMP 说！

- Header Checksum(表头检查码)
  用来检查这个 IP 表头的错误检验之用。

- Source Address
  还用讲吗？当然是来源的 IP 地址，相关的 IP 我们之前提过啰！

- Destination Address
  有来源还需要有目标才能传送，这里就是目标的 IP 地址。

- Options (其它参数)
  这个是额外的功能，提供包括安全处理机制、路由纪录、时间戳记、 严格与宽松之来源路由等。

- Padding(补齐项目)
  由于 Options 的内容不一定有多大，但是我们知道 IP 每个数据都必须要是 32 bits， 所以，若 Options 的数据不足 32 bits 时，则由 padding 主动补齐。

你只要知道 IP 表头里面还含有： TTL, Protocol, 来源 IP 与目标 IP 也就够了！ 而这个 IP 表头的来源与目标 IP ，以及那个判断通过多少路由器的 TTL ，就能了解到这个 IP 将被如何传送到目的端呐。下一节我们将介绍一下那么 IP 封包是如何被传送到目的地？

网络层之路由概念

我们在前面两个小节提到了以太网络在物理网段内可以使用 MAC 的讯框数据直接在两张网络卡之间传递数据， 那在前一节提到的 IP 相关概念里面，也知道在 Internet 的环境中，同一个网域 (Net_ID 相同的网段) 内可以直接使用广播的方式以 ARP 协议来取得 IP 与 MAC 的对应，好让我们的资料可以在同一个网域内进行传输。 好了，那问题来了，如果我想要向非同一个网域的主机要求数据呢？这个时候封包要如何传递？ 呵呵～这可就需要路由(route)的帮忙啦！这个也是在网络层 (Layer 3) 里面的重要概念喔！

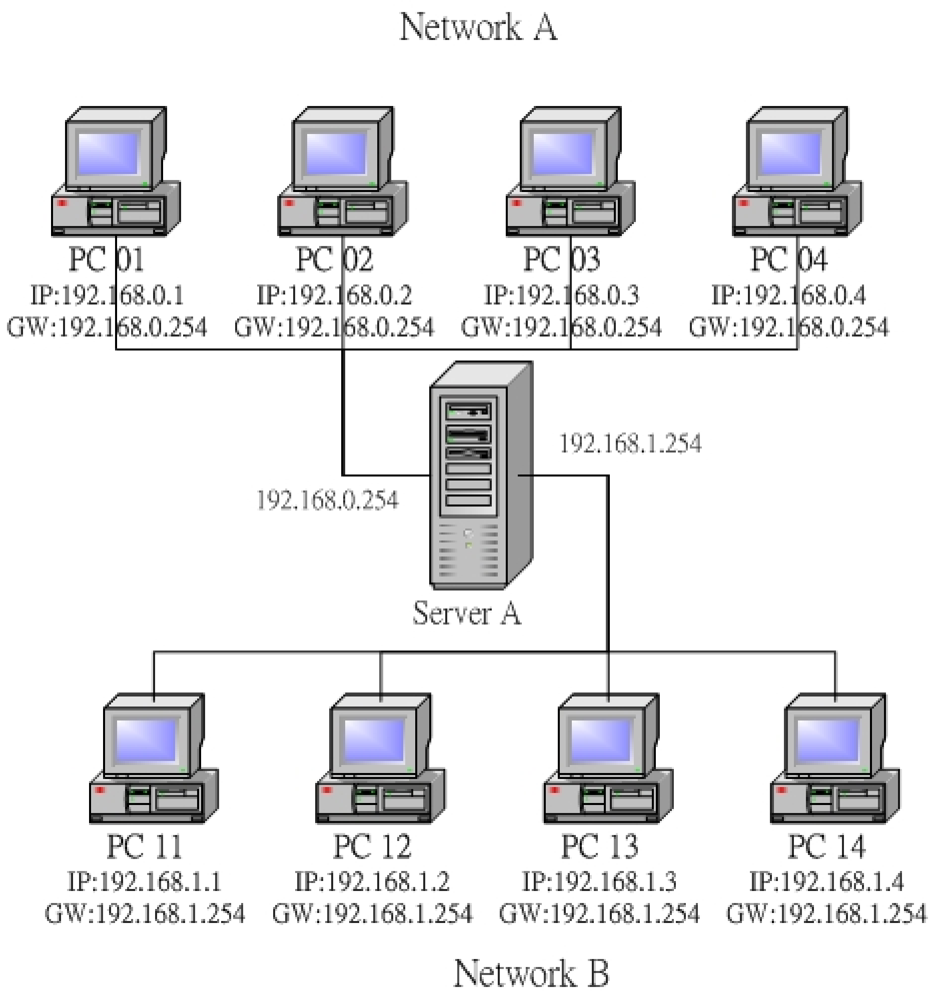
什么是路由

什么是『非同一个网域』呢？刚刚上一小节提到的 IP 应该还没有忘记吧？ 所谓的非同一个网域就是 Network/Netmask 不在同一个地址， 也就是两部主机间的 Net_ID 不相同的意思。例如参考底下的练习：

例题三：请问 192.168.10.100/25 与 192.168.10.200/25 是否在同一个网域内？ 答：

如果经过计算，会发现 192.168.10.100 的 Network 为 192.168.10.0 ，但是 192.168.10.200 的 Network 却是 192.168.10.128，由于 Net_ID 不相同，所以当然不在同一个网段内！ 关于 Network 与 Netmask 的算法则请参考上一小节。

那么万一两部不在同一个网段内的主机想要互通信息时，该如何做？ 此时就得要经过 IP 的路径选择 (routing) 功能啦!我们以下面图示的例子来做说明。下列图示当中共有两个不同的网段,分别是 Network A 与 Network B，这两个网段是经由一部路由器 (Server A) 来进行数据转递的，好了，那么当 PC01 这部主机想要传送数据到 PC11 时， 他的 IP 封包该如何传输呢？



图九、简易的路由示意图

我们知道 Network A(192.168.0.0/24) 与 Network B(192.168.1.0/24) 是不同网段， 所以 PC01 与 PC11 是不能互通资料的。不过， PC01 与 PC11 是如何知道他们两个不在同一个网段内？ 呵呵！这当然是透过 Net_ID 来发现的!那么当主机想要传送数据时,他主要的参考是啥？ 很简单!是『路由表 (route table)』,每部主机都有自己的路由表』， 让我们来看一看预设的情况下， PC01 要如何将数据传送到 PC02 呢？

1. 当 PC01 有 IP 封包需要传送时，主机会查阅 IP 封包表头的目标 IP 地址；

2. PC01 主机会分析自己的路由表，当发现目标 IP 与本机 IP 的 Net_ID 相同时(同一网域)， 则 PC01 会参考本身的 ARP 记录，而直接利用 MAC 来互相传递信息。

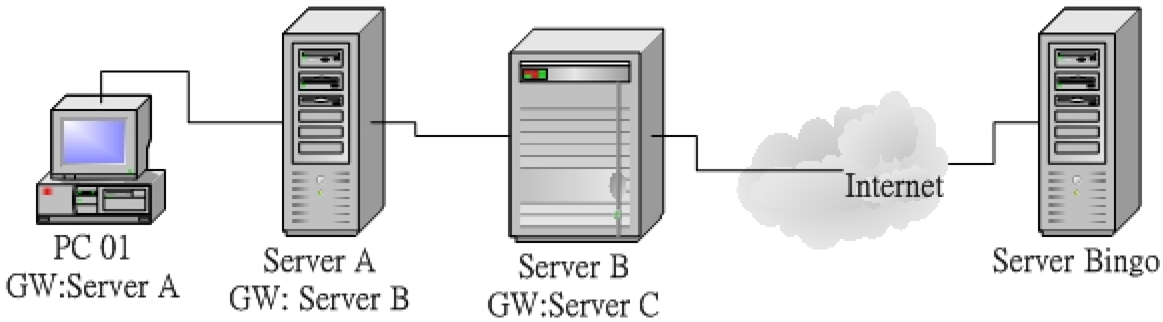
3. 但在本案例中， PC01 与 PC11 并非同一网域，因此 PC01 会分析路由表当中是否有相符合的路由设定， 如果没有的话，就直接将该 IP 封包送到预设路由器 (default gateway) 上头去， 在本案例当中 default gateway 则是 Server A 这一部。

4. 当 IP 封包被送至 Server A 之后，Server A 同样分析该 IP 封包的目标地址，然后检查 Server A 自己的路由设定，注意，通常 Server A 这个作为路由器的主机， 都会拥有两个以上的接口来沟通不同的网域的。在这个案例当中， Server A 会发现这个 IP 目标是 192.168.1.11 ，刚好是 Network B 这个相同网段的区域， 因此 Server A 会直接以 MAC 讯框将数据送给 PC11 去。

Tips:
Gateway / Router ：网关/路由器的功能就是在负责不同网域之间的封包转递 (IP Forwarder)，由于路由器具有 IP Forwarder 的功能，并且具有管理路由的能力， 所以可以将来自不同网域之间的封包进行转递的功能。此外，您的主机与您主机设定的 Gateway 必定是在同一个网段内喔！



大致的情况就是这样，所以啦，每一部主机里面都会存在着一个路由表 (Route table)，数据的传递将依据这个路由表进行传送！而一旦封包已经经由路由表的规则传送出去后， 那么主机本身就已经不再管封包的流向了，因为该封包的流向将是下一个主机 (也就是那部 Router)来进行传送，而 Router 在传送时，也是依据 Router 自己的路由表来判断该封包应该经由哪里传送出去的！例如底下的图例：



图十、路由的概念

PC 01 要将资料送到 Server Bingo 去，则依据自己的路由表，将该封包送到 Server A 去，Server A 再继续送到 Server B ，然后在一个一个的接力给他送下去，最后总是可以到达 Server Bingo 的。

当然，上面的案例是一个很简单的路由概念，事实上， Internet 上面的路由协议与变化是相当复杂的，因为 Internet 上面的路由并不是静态的，他可以随时因为环境的变化而修订每个封包的传送方向。 举例来说，数年前在新竹因为土木施工导致台湾西部整个网络缆线的中断。 不过南北的网络竟然还是能通，为什么呢？因为路由已经判断出西部缆线的终止， 因此他自动的导向台湾东部的花莲路线，虽然如此一来绕了一大圈，而且造成网络的大塞车， 不过封包还是能通就是了！这个例子仅是想告诉大家，我们上面提的路由仅是一个很简单的静态路由情况， 如果想要更深入的了解 route ，请自行参考相关书籍喔！ ^_^ 。

此外，在属于 Public 的 Internet 环境中，由于最早时的 IP 分配都已经配置妥当， 所以各单位的路由一经设定妥当后，上层的路由则无须担心啊！ IP 的分配可以参考底下的网页：

- 台湾地区 IP 核发情况：http://rms.twnic.net.tw/twnic/User/Member/Search/main7.jsp?Order=inet_aton(Startip)
- 全球 IPv4 的统计：http://www.twnic.net.tw/ipstats/ipv4stats.php

观察主机的路由

既然路由是这么的重要，而且『路由一旦设定错误， 将会造成某些封包完全无法正确的送出去！』 所以我们当然需要好好的来观察一下我们主机的路由表啦！还是请再注意一下， 每一部主机都有自己的路由表喔！观察路由表的指令很简单，就是 route ，这个指令挺难的，我们在后面章节再继续的介绍，这里仅说明一些比较简单的用法：

```
[root@linux ~]# route [-n]
参数：
-n ： 将主机名称以 IP 的方式显示
范例：

[root@linux ~]# route
ernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         192.168.0.254   0.0.0.0         UG    0      0        0 eth0

[root@linux ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         192.168.0.254   0.0.0.0         UG    0      0        0 eth0

# 上面输出的数据共有八个字段，您需要注意的有几个地方：
# Destination ：其实就是 Network 的意思；
# Gateway     ：就是该接口的 Gateway 那个 IP 啦！若为 0.0.0.0 表示不需要额外的 IP；
# Genmask     ：就是 Netmask 啦！与 Destination 组合成为一部主机或网域；
# Flags       ：共有多个旗标可以来表示该网域或主机代表的意义：
#               U：代表该路由可用；
#               G：代表该网域需要经由 Gateway 来帮忙转递；
#               H：代表该行路由为一部主机，而非一整个网域；
# Iface       ：就是 Interface (接口) 的意思。
```

在上面的例子当中，鸟哥是以 PC 01 这部主机的路由状态来进行说明。由于 PC 01 为 192.168.0.0/24 这个网域，所以主机已经建立了这个网域的路由了，那就是『 192.168.0.0 * 255.255.255.0 ... 』那一行所显示的讯息！当您下达 route 时， 屏幕上说明了这部机器上面共有三个路由规则，第一栏为『目的地的网域』，例如 192.168.0.0 就是一个网域咯，最后一栏显示的是 『要去到这个目的地要使用哪一个网络接口！』例如 eth0 就是网络卡的装置代号啦。如果我们要传送的封包在路由规则里面的

192.168.0.0/255.255.255.0 或者 127.0.0.0/255.0.0.0 里面时，因为第二栏 Gateway 为 * ，所以就会直接以后面的网络接口来传送出去，而不透过 Gateway 咯！

万一我们要传送的封包目的地 IP 不在路由规则里面，那么就会将封包传送到『default』所在的那个路由规则去，也就是 192.168.0.254 那个 Gateway 喔！所以，几乎每一部主机都会有一个 default gateway 来帮他们负责所有非网域内的封包转递！这是很重要的概念喔！^_^！ 关于更多的路由功能与设定方法，我们在后面的『简易 Router 架设』 当中会再次的提及呢！

常见的通讯协议

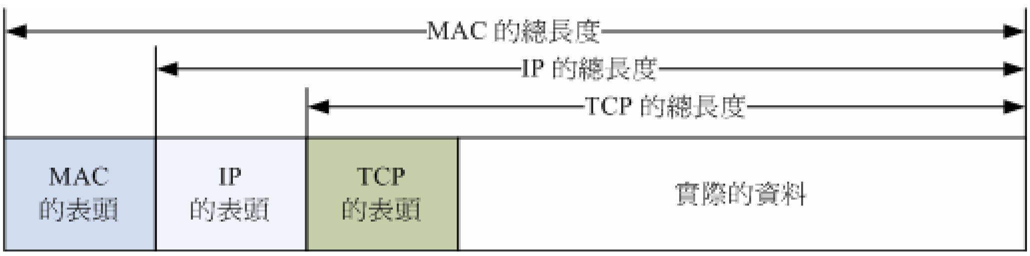
终于给他来到了封包格式的地方了！上面的咚咚大多是在网络最底层的基础知识，得自行好好的理解理解！第一次看不懂没关系，多看几次，或者是未来有些网络经验后再回来这个章节好好看一看！ 假设您已经知道了网络最底层的 IP 以及相关的参数的意义，那么应该知道的是， 网络层的协议只是提供路由的判断以确定封包的传送路径， 但是这些协议并没有管理可能由于网络媒体的损坏问题，或者是网络的负荷过重以及其它不可预期的情况， 而造成封包损毁或者被丢弃的状态。为了使封包的传送过程中更具有稳定性与可靠性， 我们就得提供一套机制来让资料可以没有错误的到达到目的地。

在 TCP/IP 这个协议组合当中，TCP (Transmission Control Protocol) 就是用来做为传送的一个协议，当然啦，还有一个 UDP 的协议呢！在 TCP 这个协议当中， 他提供了较为稳定而且可靠的联机状态，至于 UDP 则是一个比较没有这么可靠的联机型态了。 除了这个 TCP 与 UDP 之外，其它相关的网络协议请参考前一小节 IP 封包内的 Protocol 说明。 底下我们就来分别谈一谈重要的 TCP/UDP/ICMP 吧！

TCP 协定

在前几个小节内谈到的 IP 与路由的相关说明中，我们知道 IP 与路由仅能将数据封包传送到正确的目标而已， 但是这个目的地是否真的能够收下来这个封包？那可就不一定了。要确认该数据能否正确的被目的端所接收， 就必须要在数据封包上面多加一些参数来判断才行。

在前面的 OSI 七层协议当中，在网络层的 IP 之上则是传送层，而传送层的数据打包成什么？ 最常见的就是 TCP 封包了。这个 TCP 封包数据必须要能够放到 IP 的数据袋当中才行喔！ 所以，我们可以将 MAC, IP 与 TCP 的封包数据这样看：



图十一、各封包之间的相关性

所以说，IP 除了表头之外的 Data 内容其实就是 TCP 封包的表头与内容；而 MAC 的 Data 内容， 就是一个完整的 IP 封包数据！这也是我们上头提到的，最终还是得以 MAC 能够支持的最大容许容量， 才能够决定 IP 与 TCP 封包是否需要再进行分段的工作。那么既然 MAC 与 IP 都有表头数据， 想当然尔，TCP 也有表头数据来记录该封包的相关信息啰？？没错啦～ TCP 封包的表头是长这个样子的：

<table>
<tr><td>4 bits</td><td>6 bits</td><td>6 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td colspan="3">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="5">Sequence Number</td></tr>
<tr><td colspan="5">Acknowledge Number</td></tr>
<tr><td>Data Offset</td><td>Reserved</td><td>Code</td><td colspan="2">Window</td></tr>
<tr><td colspan="3">Ckecksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="3">Options</td><td colspan="2">Padding</td></tr>
<tr><td colspan="5">Data</td></tr>
</table>

图十二、TCP 封包的表头资料

上图就是一个 TCP 封包的表头数据，各个项目以 Source Port, Destination Port 及 Code 算是比较重要的项目，底下我们就分别来谈一谈各个表头数据的内容吧！

- Source Port & Destination Port (来源端口口 & 目标端口口)
  什么是埠口(port)？我们知道 IP 封包的传送主要是藉由 IP 地址连接两端，但是到底这个联机的通道是连接到哪里去呢？没错！就是连接到 port 上头啦！ 举例来说，鸟站 (http://linux.vbird.org) 有开放 WWW 服务器， 这表示鸟站的主机必须要启动一个可以让 client 端连接的端口，这个端口就是 port ， 中文翻译成为埠口。同样的，客户端想要连接到鸟哥的鸟站时，就必须要在 client 主机上面启动一个 port ，这样这两个主机才能够利用这条『通道』来传递封包数据喔！ 这个目标与来源 port 的纪录，可以说是 TCP 封包上最重要的参数了！ 下个小单元我们还会继续介绍。

- Sequence Number (封包序号)
  由于 TCP 封包必须要带入 IP 封包当中，所以如果 TCP 数据太大时(大于 IP 封包的容许程度)，就得要进行分段。这个 Sequence Number 就是记录每个封包的序号， 可以让收受端重新将 TCP 的数据组合起来。

- Acknowledge Number (回应序号)
  为了确认主机端确实有收到我们 client 端所送出的封包数据，我们 client 端当然希望能够收到主机方面的响应，那就是这个 Acknowledge Number 的用途了。 当 client 端收到这个确认码时，就能够确定之前传递的封包已经被正确的收下了。

- Data Offset (资料补偿)
  在图十二倒数第二行有个 Options 字段对吧！那个 Options 的字段长度是非固定的， 而为了要确认整个 TCP 封包的大小，就需要这个标志来说明整个封包区段的起始位置。

- Reserved (保留)
  未使用的保留字段。

- Code (Control Flag, 控制标志码)
  当我们在进行网络联机的时候，必须要说明这个联机的状态，好让接收端了解这个封包的主要动

作。 这可是一个非常重要的句柄喔！这个字段共有 6 个 bits ，分别代表 6 个句柄，若为 1 则为启动。分别说明如下：

- URG(Urgent)：若为 1 则代表该封包为紧急封包， 接收端应该要紧急处理，且图十二当中的 Urgent Pointer 字段也会被启用。
- ACK(Acknowledge)：若为 1 代表这个封包为响应封包， 则与上面提到的 Acknowledge Number 有关。
- PSH(Push function)：若为 1 时， 代表要求对方立即传送缓冲区内的其它对应封包，而无须等待缓冲区满了才送。
- RST(Reset)：如果 RST 为 1 的时候， 表示联机会被马上结束，而无需等待终止确认手续。这也就是说，这是个强制结束的联机， 且发送端已断线。
- SYN(Synchronous)：若为 1 ， 表示发送端希望双方建立同步处理，也就是要求建立联机。通常带有 SYN 标志的封包表示『主动』要连接到对方的意思。
- FIN(Finish)：若为 1 ，表示传送结束， 所以通知对方数据传毕，是否同意断线，只是发送者还在等待对方的响应而已。

其中比较常见到的应该是 ACK/SYN/FIN 等，这三个句柄是务必要记下来的， 这样未来在谈到防火墙的时候，您才会比较清楚为啥每个 TCP 封包都有所谓的『状态』条件！ 那就是因为联机方向的不同所致啊！底下我们会进一步讨论喔！

- Window (滑动窗口)
  主要是用来控制封包的流量的，可以告知对方目前本身有的缓冲器容量(Receive Buffer) 还可以接收封包。当 Window=0 时，代表缓冲器已经额满，所以应该要暂停传输数据。 Window 的单位是 byte。

- Checksum(确认检查码)
  当数据要由发送端送出前，会进行一个检验的动作，并将该动作的检验值标注在这个字段上； 而接收者收到这个封包之后，会再次的对封包进行验证，并且比对原发送的 Checksum 值是否相符，如果相符就接受，若不符就会假设该封包已经损毁，进而要求对方重新发送此封包！

- Urgent Pointer(紧急资料)
  这个字段是在 Code 字段内的 URG = 1 时才会产生作用。可以告知紧急数据所在的位置。

- Options(任意资料)
  目前此字段仅应用于表示接收端可以接收的最大数据区段容量，若此字段不使用， 表示可以使用任意数据区段的大小。这个字段较少使用。

- Padding(补足字段)
  如同 IP 封包需要有固定的 32bits 表头一样， Options 由于字段为非固定， 所以也需要 Padding 字段来加以补齐才行。同样也是 32 bits 的整数。

通讯端口口与 Socket

在刚刚上头提到的 TCP 表头数据后，您大概也清楚了要建立一个 TCP 封包时所需要检验的相关参数可不少啊！其中最重要的就属通讯端口口 (port) 了。 这个 port 主要是由主机的程序所触发的，网络上面的其它 client 端， 可以通过这个埠口直接与启动该 port 的程序相互沟通， 而达到数据传输的目的。我们都知道二进制程序 (binary program) 才是真的主机认识的程序指令， 那么我们要启动网络服务时，其实也就是启动一个 program 就是了。但是网络上面如何与您的 program 互通数据呢？就是透过在网络接口上面的 port 来达成的啦。

不过，必须要注意的是 port 的沟通是双向的， 举例来说，当我们要使用浏览器连接到奇摩雅虎查阅数据时， 我们必须要使用 client 端主机的浏览器连接到 Yahoo 主机的 WWW 服务器软件上面。 由于是透过网络接口，所以我们的浏览器也必须要启动一个 port 并且透过这个 port 连接到 Yahoo 主机的 WWW port 上头去，然后透过 TCP 封包上头各项参数的确认后， 才能够建立联机，并进一步开始传输数据啊！

现在来想一想，我们的主机上面有多少网络接口的 port 可用呢？ 基本上就有 65536 (2^{16})。那我们联机到 Yahoo 时， 是联机到 Yahoo 主机的那个 port 啊？如果不知道几号 port ，那又该如何建立起这个联机呢？ 所以啰，Internet 上面已经有很多规范好的固定 port (well-known port) 在提供使用者建立服务器时启用的 port number 啦。这些 port number 通常小于 1024 ， 且是提供给许多知名的网络服务软件用的。在我们的 Linux 环境下， 各网络服务与 port number 的对应预设给他写在 /etc/services 档案内喔！ 不过如果是 client 端的话，由于 client 端都是主动向 server 端要数据， 所以 client 端的 port number 就使用随机取一个大于 1024 以上且没有在用的 port number 来进行联机了。底下鸟哥列出几个常见的 port number 与网络服务的对应：

连接埠口	服务名称与内容
20	FTP-data，档案传输协议所使用的主动数据传输端口口
21	FTP，档案传输协议的命令通道
22	SSH，较为安全的远程联机服务器
23	Telnet，早期的远程联机服务器软件
25	SMTP，简单邮件传递协议，用在作为 mail server 的埠口
53	DNS，用在作为名称解析的领域名称服务器
80	WWW，这个重要吧！就是全球信息网服务器
110	POP3，邮件收信协议，办公室用的收信软件都是透过他
443	https，有安全加密机制的 WWW 服务器

另外一点比较值得注意的是，小于 1023 以下的埠口要启动时， 启动者的身份必须要是 root 才行！这个限制挺重要的，大家不要忘记了喔！

曾经有一个朋友问过我说：『一部主机上面这么多服务，那我们跟这部主机进行联机时，该主机怎么知道我们要的数据是 WWW 还是 FTP 啊？！』呵呵！这就是 port 的不同的结果啦！因为每种 Client 软件他们所需要的数据都不相同，例如上面提到的浏览器(Netscape 以及 IE)

他们所需要的数据是 WWW ，所以该软件预设就会向主机的 80 port 索求数据；而如果您是使用 cuteftp 来进行与主机的 FTP 数据索求时， cuteftp 这个 Client 软件当然预设就是向主机的 FTP 相关端口口（ 预设就是 port 21 ）进行连接的动作啦！所以当然就可以正确无误的取得 Client 端所需要的数据了

举个例子来说，一部主机就好像是一间多功能银行，该银行内的每个负责不同业务的窗口就好像是通讯端口口，而我们民众就好像是 Client 端来的封包。当您进入银行想要缴纳信用卡账单时，一到门口服务人员就会指示您直接到该窗口去缴纳，当然，如果您是要领钱，服务人员就会请您到领钱的窗口去填写数据，您是不会跑错的对吧！ ^_^。万一跑错了怎么办？呵呵！当然该窗口就会告诉您『我不负责这个业务，您请回去！』，呵呵！所以该次的联机就会『无法成功』咯！

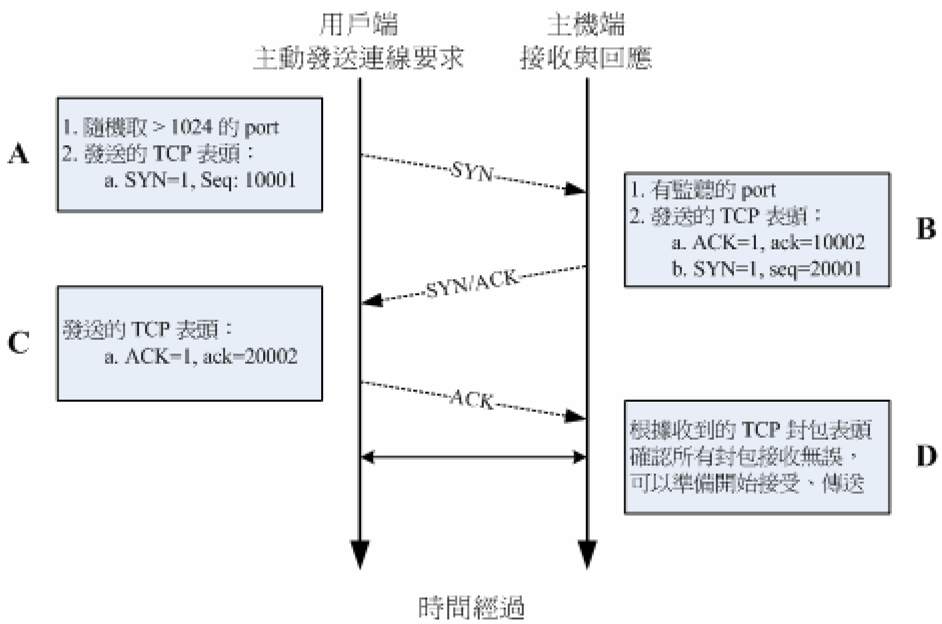
谈过了通讯端口口后，再来聊一聊每个封包的基本内容有哪些数据呢？这就得要谈论到 Socket Pair（ 成对的端口 ）了！当本机发送出封包时，主机是根据 IP 封包内的 IP 地址来判别来源与目的地的行进路线，并且，也需要 TCP 封包内的 port number 来告知 Client 与 Server 是以哪一个埠口来进行联机的对吧！所以呢，一个联机过程所包含的底下这些咚咚就称为成对插槽 (socket pair) 了：

- 来源 IP（ Source Address ）
- 目的 IP（ Destination Address ）
- 来源埠口 （ Source Port ）
- 目的埠口 （ Destination Port ）

这四个封包的基本信息是相当重要的！您得必须要了解喔！

封包的传送
OK，从前面这样一路看下来，相信您对于 OSI 七层协议当中的 MAC 讯框与 IP 封包有一定程度的了解了，也知道七层协议必须要在不同的主机之间一再地拿出来察看，因为 Internet 是用 IP 拿传递封包数据，而实体线路则是使用 MAC 讯框。那我们也了解 TCP 封包的表头数据后，再来就是要了解一下， 那我如何利用 TCP 这个传送层的协议来进行实际的封包接收呢？ 当然是得要透过 TCP 表头的 Sequence Number 来组合收集大的 TCP 封包， 也必须要透过 Code (Control Flags) 来了解到这个封包的特性才行。 说穿了，我们可以使用信封带来说明，实际的内容 (data) 是在信封袋当中的， 而信封的外面的信息就是各个封包的表头数据啦！

那么如何藉由 TCP 的表头来确认这个封包有实际被对方接收，并进一部与对方主机达成联机？ 我们以底下的图示来作为说明。



图十三、封包连接模式之三向交握

在上面的封包连接模式当中，在建立联机之前都必须要通过三个确认的动作， 所以这种联机方式也就被称为三向交握(Three-way handshake)。 那么我们将整个流程依据上面的 A, B, C, D 四个阶段来说明一下：

- A:封包发起
  当客户端想要对服务器端联机时，就必须要送出一个要求联机的封包， 此时客户端必须随机取用一个大于 1024 以上的端口口来做为程序沟通的接口。然后在 TCP 的表头当中，必须要带有 SYN 的主动联机(SYN=1)，并且记下发送出联机封包给服务器端的序号 (Sequence number = 10001) 。

- B:封包接收与确认封包传送
  当服务器接到这个封包，并且确定要接收这个封包后，就会开始制作一个同时带有 SYN=1, ACK=1 的封包， 其中那个 acknowledge 的号码是要给 client 端确认用的，所以该数字会比(A 步骤) 里面的 Sequence 号码多一号 (ack = 10001+1 = 10002)， 那我们服务器也必须要确认客户端确实可以接收我们的封包才行，所以也会发送出一个 Sequence (seq=20001) 给客户端，并且开始等待客户端给我们服务器端的回应喔！

- C:回送确认封包
  当客户端收到来自服务器端的 ACK 数字后 (10002) 就能够确认之前那个要求封包被正确的收受了，接下来如果客户端也同意与服务器端建立联机时，就会再次的发送一个确认封包 (ACK=1) 给服务器，亦即是 acknowledge = 20001+1 = 20002 啰。

- D:取得最后确认
  若一切都顺利，在服务器端收到带有 ACK=1 且 ack=20002 序号的封包后，就能够建立起这次的联机了。

也就是说，你必须要了解『网络是双向的』这个事实！ 所以不论是服务器端还是客户端，都必须要透过一次 SYN 与 ACK 来建立联机， 所以总共会进行三次的交谈！ 在设定防火墙或者是追踪网络联机的问题时，这个『双向』的概念最容易被忽略， 而常常导致无法联机成功的问题啊！切记切记！

在建立了联机之后，该次联机通道就可以在客户端与服务器端建立起一对 socket pair ， 然后通过该 socket pair 进行 TCP 封包的 PSH、FIN 等数据传输与联机中断等动作啰！

UDP 协定

UDP 的全名是：『User Datagram Protocol, 用户数据流协议』，UDP 与 TCP 不一样， UDP 不提供可靠的传输模式，因为他不是联机导向的一个机制，这是因为在 UDP 的传送过程中，接受端在接受到封包之后，不会回复响应封包 (ACK) 给发送端，所以封包并没有像 TCP 封包有较为严密的验证机制。至于 UDP 的表头资料如下表所示：

16 bytes	16 bytes
Source Port	Destination Port
Message Length	Checksum
Data	

图十三、UDP 封包的表头资料

TCP 封包确实是比较可靠的，因为通过三向交握嘛！不过，也由于三向交握的缘故， TCP 封包的传输速度会较慢。 至于 UDP 封包由于不需要确认对方是否有正确的收到数据，故表头数据较少，也因为如此所以 UDP 就可以在 Data 处填入更多的数据了。同时 UDP 比较适合需要实时反应的一些数据流，例如实时通讯软件或者是影像实时传送软件等， 就可以使用这类的封包传送说。也就是说， UDP 传输协议并不考虑联机要求、联机终止与流量控制等特性， 所以使用的时机是当数据的正确性不很重要时，例如上面提到的实时通讯软件啊！

另外，很多的软件其实是同时提供 TCP 与 UDP 的传输协议的，举例来说，查询主机名称的 DNS 服务就同时提供了 UDP/TCP 协议。由于 UDP 较为快速，所以我们 client 端可以先使用 UDP 来与服务器联机。 但是当使用 UDP 联机却还是无法取得正确的数据时，便转换为较为可靠的 TCP 传输协议来进行数据的传输啰。 这样可以同时兼顾快速与可靠的传输说！

ICMP 协定

ICMP 的全称是『 Internet Control Message Protocol, 因特网讯息控制协议 』。 基本上，ICMP 是一个错误侦测与回报的机制，最大的功能就是可以确保我们网络的联机状态与联机的正确性！ 同样的，ICMP 封包也是必须要装在 IP 封包的 Data 内才行喔！ 因为在 Internet 上面有传输能力的就是 IP 封包啦！ ICMP 有相当多的类别可以侦测与回报，底下是比较常见的几个 ICMP 的类别 (Type)：

类别代号	类别名称与意义
0	Echo Reply (代表一个响应信息)
3	Distination Unreachable (表示目的地不可到达)
4	Source Quench (当 router 的负载过高时，此类别码可用来让发送端停止发送讯息)

5	Redirect (用来重新导向路由路径的信息)
8	Echo Request (请求响应讯息)
11	Time Exceeded for a Datagram (当数据封包在某些路由传送的现象中造成逾时状态，此类别码可告知来源该封包已被忽略的讯息)
12	Parameter Problem on a Datagram (当一个 ICMP 封包重复之前的错误时，会回复来源主机关于参数错误的讯息)
13	Timestamp Request (要求对方送出时间讯息，用以计算路由时间的差异，以满足同步性协议的要求)
14	Timestamp Replay (此讯息纯粹是响应 Timestamp Request 用的)
15	Information Request (在 RARP 协议应用之前，此讯息是用来在开机时取得网络信息)
16	Information Reply (用以响应 Infromation Request 讯息)
17	Address Mask Request (这讯息是用来查询子网络 mask 设定信息)
18	Address Mask Reply (响应子网络 mask 查询讯息的)

那么我们是如何利用 ICMP 来检验网络的状态呢？最简单的指令就是 ping 与 traceroute 了，这两个指令可以透过 ICMP 封包的辅助来确认与回报网络主机的状态。在设定防火墙的时候， 我们最容易忽略的就是这个 ICMP 的封包了，因为只会记住 TCP/UDP 而已～事实上， ICMP 封包可以帮助联机的状态回报，除了上述的 8 可以考虑关闭之外，基本上， ICMP 封包也不应该全部都挡掉喔！

MTU 的限制

我们在本章的第一部份谈到 MAC 时，不是有提到标准以太网络封包的信息容量大约在 1500byes 吗？ 所以 IP 封包、TCP 封包以及其它相关的封包，如果能够控制在 1500 bytes 内的话， 那么 IP 封包将不需要重组成可以放进 MAC 讯框的分段了！

让我们来想一个小案例，假设你的数据量大到 60000 bytes 好了，如果你使用的传输协议为 TCP 封包，万一你没有考虑到 MAC 讯框的大小，而让整笔数据放到可以容纳最大 65535 bytes 的 IP 封包内， 此时 IP 封包已经建立好成为内含 60000 bytes 的数据包了，但是再往下到达数据连接层时，唉～ 这个 IP 封包就得要进行重组，好让 IP 封包可以放到 MAC 讯框当中！您说，这个时候不是又得让系统多进行一段手续，而导致网络效能的低落吗？

此时，如果能够规范 TCP 以及 IP 在包起来时就考虑讯框的最大容量时， 不就可以减少很多数据重组的问题啰？呵呵呵呵！没错啊！ 那就是最大传输单元 (Maximum Transmission Unit, MTU) 这个设定值的重要性啊！

一般来说，我们的 Gigabit 网络卡已经可以支持 Jumbo frame ，所以 MTU 值都可以到达 9000 bytes 左右， 不过，不建议您设定 MTU 成为 9000 喔！为什么呢？因为我们的封包总是需要在 Internet 上面跑吧？ 您无法确认所有的网络媒体都是支持那么大的 MTU 对吧！ 如果您的 9000 bytes 封包通过一个不支持 Jumbo frame 的网络媒体时， 好一点的是该网络媒体 (例如 router) 会主动的帮您重组封包而进行传送，差一点的可能就直接回报这个封包无效而丢弃了～那个时候可就糗大啰～ 所以，MTU 设定为 9000

这种事情，大概仅能在内部网络的环境作作～ 举例来说，很多的内部丛集系统 (cluster) 就将他们的内部网络环境 MTU 设定为 9000， 但是对外的适配卡可还是原本的标准 1500 喔！ ^_^

也就是说，不论您的网络媒体支持 MTU 到多大，您必须要考虑到您的封包需要传到目的地时， 所需要经过的所有网络媒体，然后再来决定您的 MTU 设定才行。

> Tips:
> 事实上，MTU 不会刚好等于 1500 呐！这是因为不论是 IP 封包或者是 TCP 封包都会有表头数据，这些表头数据都会占用去一些位容量，所以 MTU 就会比标准以太网络容量的 1500 小一些。



封包过滤的防火墙概念

由上面的说明当中，我们知道数据的传送其实就是封包的发出与接受的动作啦！并且不同的封包上面都有不一样的表头 (header)，此外，封包上面通常都会具有四个基本的信息，那就是 socket pair 里面提到的『来源与目的 IP 以及来源与目的端的 port number』 。当然啦，如果是可靠性联机的 TCP 封包，还包含 Control Flag 里面的 SYN/ACK 等等重要的信息呢！好了，开始动一动脑筋， 有没有想到『网络防火墙』的字眼啊？ 网络防火墙可以抵挡掉一些可能有问题的封包，那么在 Linux 系统上面是怎么挡掉封包的呢？其实说来也是很简单，既然封包的表头上面已经有这么多的重要信息， 那么我就利用一些防火墙机制与软件来进行封包表头的分析，并且设定分析的规则，当发现某些特定的 IP 、特定的埠口或者是特定的封包信息(SYN/ACK等等)，那么就将该封包给他丢弃， 那就是最基本的防火墙原理了！

举例来说，大家都知道 Telnet 这个服务器是挺危险的，而 Telnet 使用的 port number 为 23 ，所以，当我们使用软件去分析要送进我们主机的封包时， 只要发现该封包的目的地是我们主机的 port 23 ，就将该封包丢掉去！那就是最基本的防火墙案例啦！ 更多的防火墙信息我们会在后头的『简易防火墙』与 『认识网络安全』当中进行更多的说明喔！

连上 Internet 前的准备事项

讲了这么多，其实我们最需要的仅是『连接上 Internet 』啦！那么在 Internet 上面其实使用的是 TCP/IP 这个通讯协议，所以我们就需要 Public IP 来连接上 Internet 啊！您说对吧～ 不过，您有没有发现一件事，那就是『为啥我不知道 Yahoo 的主机 IP ，但是俺的主机却可以连到 Yahoo 主机上？』 如果您有发现这个问题的话，哈哈！您可以准备开始设定网络啰～ ^_^

什么是主机名称与 DNS

除了上面提到的最基本的网络基础概念之外，这里还必须要先谈一个基本的观念， 否则后续的主机名称查询设定挺难说明白的！好了，我们知道计算机在网络上面要找寻主机的时后，是利用 IP 来寻址，而以 TCP/UDP/ICMP 等数据来进行传送的，并且传送的过程中还会去检验封包的信息。 总归一句话，网络是靠 TCP/IP 家族来达成的，所以必须要知道 IP 之后，计算机才能够连上网络以及传送数据。

问题是，计算机网络是依据人类的需要来建立的，不过人类对于 IP 这一类的数字并不具有敏感性，即使 IP 已经被简化为十进制了，但是人类就是对数字没有办法啊！怎么办？没关系，反正计算机都有主机名称嘛！

那么我就将主机名称与他的 IP 对应起来，未来要连接上该计算机时，只要知道该计算机的主机名称就好了，因为 IP 已经对应到主机名称了嘛！所以人类也容易记忆文字类的主机名称，计算机也可以藉由对应来找到他必须要知道的 IP ，啊！真是皆大欢喜啊！

这个主机名称 (Hostname) 对应 IP 的系统，就是鼎鼎有名的 Domain Name System (DNS) 咯！也就是说，DNS 这个服务的最大功能就是在进行『主机名称与该主机的 IP 的对应』的一项协议。 DNS 在网络环境当中是相当常被使用到的一项协议喔！举个例子来说，像鸟哥我常常会连到奇摩雅虎的 WWW 网站去看最新的新闻，那么我一定需要将奇摩雅虎的 WWW 网站的 IP 背下来吗？！天呐， 鸟哥的忘性这么好，怎么可能将 IP 背下来？！不过，如果是要将奇摩站的主机名称背下来的话， 那就容易的多了！不就是 http://tw.yahoo.com 吗？！而既然计算机主机只认识 IP 而已， 因此当我在浏览器上面输入了『http://tw.yahoo.com』的时后，我的计算机首先就会藉由向 DNS 主机查询 tw.yahoo.com 的 IP 后，再将查询到的 IP 结果回应给我的浏览器， 那么我的浏览器就可以藉由该 IP 来连接上主机啦！

发现了吗？我的计算机必须要向 DNS 主机查询 Hostname 对应 IP 的信息 喔！那么那部 DNS 主机的 IP 就必须要在我的计算机里面设定好才行，并且必须要是输入 IP 喔，不然我的计算机怎么连到 DNS 主机去要求数据呢？呵呵！在 Linux 里面，DNS 主机 IP 的设定就是在 /etc/resolv.conf 这个档案里面啦！

目前各大 ISP 都有提供他们的 DNS 主机 IP 给他们的用户，好设定客户自己计算机的 DNS 查询主机，不过，如果您忘记了或者是您使用的环境中并没有提供 DNS 主机呢？呵呵！没有关系， 那就设定 Hinet 那个最大的 DNS 主机吧！ IP 是 168.95.1.1 咯！要设定好 DNS 之后，未来上网浏览时，才能使用主机名称喔！不然就得一定需要使用 IP 才能上网呢！DNS 是很重要的，他的原理也顶复杂的，更详细的原理我们在后面的 『DNS 服务器架设』里面进行更多更详细的说明喔！这里仅提个大纲！

一组可以连上 Internet 的必要网络参数

从上面的所有说明当中，我们知道一部主机要能够使用网络，必须要有 IP ，而 IP 的设定当中，就必须要有 IP, Network, Broadcast, Netmask 等参数，此外，还需要考虑到路由里面的 Default Gateway 才能够正确的将非同网域的封包给他传送出去。 此外，考虑到主机名称与 IP 的对应，所以您还必须要给予系统一个 DNS 主机的 IP 才行～ 所以说，一组合理的网络设定需要哪些数据呢？呵呵！就是：

- IP
- Netmask
- Network
- Broadcast
- Gateway
- DNS

没错！就是这些数据！如果您是使用 ADSL 拨接来上网的话，上面这些数据都是由 ISP 直接给您的，那您只要使用拨接程序进行拨接到 ISP 的工作之后， 这些数据就自动的在您的主机上面设定完成了！但是如果是固定制 (如学术网络) 的话，那么就得自行使用上面的参数来设定您的主机啰！缺一不可呢！以 192.168.0.0/24 这个 C Class 为例的话，那么您就必须要在您的主机上面设定好底下的参数：

- IP: 由 192.168.0~192.168.0.254
- Netmask: 255.255.255.0

- Network: 192.168.0.0
- Broadcast: 192.168.0.255
- Gateway: 每个环境都不同，请自行询问网络管理员
- DNS: 也可以直接设定成 168.95.1.1

重点回顾：

- 只要是能够连接上 Internet 的主机，都有危险，不要以为小网站就不会被 cracker 所破解；
- 虽然目前的网络媒体多以以太网络为标准，但网络媒体不只有以太网络而已；
- Internet 主要是由 Internet Network Information Center (INTERNIC) 所维护，但其仅维护一些技术文件的推展；
- 以太网络的标准相当多，速度的定义亦不相同，购买时需要特别留意其速度标准。
- 以太网络的 RJ-45 网络线，由于 568A/568B 接头的不同而又分为并行线与跳线；
- 网络媒体都有其最大可接受的封包量，在以太网络上可接收的封包为 MAC (Media Access Control) 讯框
- 以太网络上最重要的传输数据为 Carrier Sence Multiple Access with Collision Detect (CSMA/CD) 技术， 至于传输过程当中，最重要的 MAC 讯框内以硬件地址 (hardware address) 数据最为重要；
- 以太网络的交换技术 (switch) 已经可以克服 CSMA/CD 所发生的封包碰撞情况，因为 switch 为非共享媒体
- 透过八蕊的网络线 (Cat 5 以上等级)，现在的以太网络可以支持全双工模式；
- OSI 七层协议为一个网络模型 (model) ，并非硬性规定。这七层协议可以协助软硬件开发有一个基本的准则， 且每一分层各自独立，方便使用者开发；
- 现今的网络基础是架构在 TCP/IP 这个通讯协议上面；
- 数据连接层里重要的信息为 MAC (Media Access Control)，亦可称为硬件地址，而 ARP Table 可以用来对应 MAC 与软件地址 (IP) ；
- 网络的传输单位使用 bit 而不是 byte ；
- 在网络媒体方面， Hub 为共享媒体，因此可能会有封包碰撞的问题，至于 Switch 由于加入了 switch port 与 MAC 的对应，因此已经克服了封包碰撞的问题，也就是说，Switch 并不是共享媒体；
- IP 为 32 bits 所组成的，为了适应人类的记忆，因此转成四组十进制的数据；
- IP 主要分为 Host ID 与 Net ID 两部份，加上 Netmask 这个参数后，可以设定『网域』的概念；
- 所谓的同一网域指的是 Net_ID 相同，但 Host_ID 不同的环境下；
- 根据 IP 网域的大小，可将 IP 的等级分为 A, B, C 三种常见的等级；
- Loopback 这个网段在 127.0.0.0/8 ，用在每个操作系统内部的循环测试中。
- 网域可继续分成更小的网域 (subnetwork)，主要是透过将 Host_ID 借位成为 Net_ID 的技术；
- 若 IP 封包大于 MAC 可接受的最大值时，就得将该 IP 封包重组分段；
- IP 只有两种，就是 Public IP 与 Private IP ，中文应该翻译为 公共 IP 与 私有(或保留) IP，私有 IP 与私有路由不可以直接连接到 Internet 上；
- 每一部主机都有自己的路由表，这个路由表规定了封包的传送途径，在路由表当中，最重要者为预设的通讯闸 (Gateway/Router)；
- TCP 协议的表头数据当中，那个 Code (control flags) 所带有的 ACK, SYN, FIN 等为常见的旗标， 可以控制封包的联机成功与否；

- TCP 与 IP 的 IP address/Port 可以组成一对 socket pair
- 网络联机都是双向的，在 TCP 的连向当中，需要进行客户端与服务器端两次的 SYN/ACK 封包发送与确认， 所以一次 TCP 联机确认时，需要进行三向交握的流程；
- UDP 通讯协议由于不需要联机确认，因此适用于快速实时传输且不需要数据可靠的软件中，例如实时通讯；
- ICMP 封包最主要的功能在回报之侦测网络的状况，故不要使用防火墙将他完全挡掉；
- MTU 可以用来规范各种封包打包时的最大单位，单位为 byte。
- 一般来说，一部主机里面的网络参数应该具备有：IP, Netmask, Network, Broadcast, Gateway 等；
- 目前常见的数据封包格式有 TCP/UDP/ICMP 等，TCP 为较可靠的封包格式， 透过多种确认手段来使封包可以准确的到达目的地，至于 UDP 则略过这些确认手续，因此传送速度较快。
- 在主机的 port 当中，只有 root 可以启用小于 1023 以下的 port ；
- DNS 主要的目的在于进行 Hostname 对应 IP 的功能；

课后练习：

- 请简述 OSI 网络七层协议每一层的功能；

  请参考本章第一节的相关内容。

- 在 ISP 提供的网络服务中，他们提到传输速度为 1.5M/382K ，请问这个数据的单位为何？

  数据单位为 bits/second, 与惯用的 bytes 差 8 倍。

- 什么是 MAC (Media Access Control) ，MAC 主要的功能是什么？

  Media Access Control 的缩写，为以太网络硬件讯框的规格，以太网络就是以 MAC 讯框进行数据的传送。 目前 MAC 也常被用为以太网络卡卡号的代称。

- 什么是封包碰撞？为什么会发生封包碰撞？

  当主机要使用网络时，必须要先进行 CSMA/CD 监听网络，如果(1)网络使用频繁 (2)网络间隔太大， 则可能会发生监听时均显示无主机使用，但发出封包后却发生同步发送封包的情况，此时两个封包就会产生碰撞， 造成数据损毁。

- ARP Table 的作用为何？如何在我的 Linux 察看我的 ARP 表格？

  ARP 协议主要在分析 MAC 与 IP 的对应，而解析完毕后的数据会存在系统的内存中， 下次要传送到相同的 IP 时，就会主动的直接以该 MAC 传送，而不发送广播封包询问整个网域了。
  利用 arp -n 即可

- 简略说明 Netmask 的作用与优点；

Netmask 可以用来区分网域，且 Netmask 可以有效的增加网络的效率，这是因为 Netmask 可以定义出一个网域的大小，那么 broadcast 的时间就可以降低很多！一般来说， 我们如果要将一个大网域再细分为小网域，也需要藉由 Netmask 来进行 subnet 的切割。

- 我有一组网域为：192.168.0.0/28 ，请问这个网域的 Network, Netmask, Broadcast 各为多少？而可以使用的 IP 数量与范围各是多少？

  因为共有 28 个 bits 是不可动的，所以 Netmask 地址的最后一个数字为 11110000，也就是 (128+64+32+16=240) ，所以：
  Network：192.168.0.0
  Netmask：255.255.255.240
  Broadcast：192.168.0.15
  IP：由 192.168.0.1 ~ 192.168.0.14 共 14 个可用 IP 喔！

- 承上题，如果网域是 192.168.0.128/29 呢？

  因为是 29 个 bits 不可动，所以最后一个 Netmask 的地址为： 11111000 也就是 (128+64+32+16+8=248)，所以：
  Network：192.168.0.128
  Netmask：255.255.255.248
  Broadcast：192.168.0.135
  IP：由 192.168.0.129 ~ 192.168.0.134 共 6 个可用的 IP 喔！

- 我要将 192.168.100.0/24 这个 C Class 的网域分为 4 个子网域，请问这四个子网域要如何表示？

  既然要分为四个网域，也就是还需要藉助 Netmask 的两个 bits (2 的 2 次方为 4 啊！)，所以 Netmask 会变成 255.255.255.192 ，每个子网域会有 256/4=64 个 IP ，而必须要扣除 Network 与 Broadcast ，所以每个子网域会有 62 个可用 IP 喔！因此，四个子网域的表示方法为：192.168.100.0/26, 192.168.100.64/26, 192.168.100.128/26, 192.168.100.192/26。

- 如何观察 Linux 主机上面的路由信息 (route table)？

  路由信息的观察可以下达 route 来直接察看！或者是下达 route -n 亦可

- TCP 封包上面的 SYN 与 ACK 标志代表的意义为何？

  SYN 代表该封包为该系列联机的第一个封包，亦即是主动联机的意思；
  ACK 则代表该封包为确认封包，亦即是回应封包！

- 什么是三向交握？在哪一种封包格式上面才会有三向交握？

  使用 TCP 封包才会有三向交握。TCP 封包的三向交握是一个确认封包正确性的重要步骤，通过 SYN, SYN/ACK, ACK 三个封包的确认无误后，才能够建立联机。至于 UDP 封包则没有三向交握喔！

- 参考合勤科技网页上的说明 (http://www.zyxel.com.tw/product/category.php?indexFlagvalue=1028014886) 试说明何谓有网管？无网管的 switch ？此外，这些 switch 的硬件应算在 OSI 七层协议的第几层？

  有网管者，会在 switch 内部加入其它的小型 OS，藉以控管 IP 或 MAC 的流通； 通常基础的 switch 仅达控管 MAC ，故为 OSI 第二层(数据连接层)

- 为何 ISP 有时后会谈到『申请固定 8 个 IP ，其中只有 5 个可以用』，你觉得问题出在哪里？如果以网域的观念来看，他的 netmask 会是多少？

  因为如果是一个网域的话，那么八个 IP 前后(Host_ID 全为 0 与 1 的条件)为 Network 及 Broadcast ， 加上一个在 ISP 处的 Gateway ，所以仅有 5 个可以用。因为有 8 个 IP ，所以其 netmask 后八 bits 为 11111000 ，故为 255.255.255.248。

- Internet 协议中共包含 "Network Access Layer", "Internet Layer", "Transport Layer", "Application Layer"， 请将这四层与 OSI 七层协议的内容进行连结 (自行上网查询相关文章说明)；

  Network Access Layer: 涵盖 Data-Link 及 Physical Layer
  Internet Layer: 也是 Network Layer
  Transport Layer: 也是 Transport Layer
  Application Layer: 涵盖 Application Layer, Persentatin Layer, Session Layer.

- 请自行上网查询关于 NetBIOS 这个通讯协议的相关理论基础，并请说明 NetBIOS 是否可以跨路由？

  请自行参考网中人的网络基础文章

- 什么是 Socket pair ？包含哪些基本数据？

  由 IP 封包的 IP address 与 TCP 封包的 port number 达成，分别为目的端的 IP/port 与本地端的 IP/port。

- 分别说明 568A 及 568B 这两种 RJ-45 接头的蕊线颜色排列顺序；

  568A: 白绿　绿　白橙　蓝　白蓝　橙　白棕　棕
  568B:白橙　橙　白绿　蓝　白蓝　绿　白棕　棕

- IP 有一段 A Class 的网段分给系统做为测试用，请问该网段为？设定的名称为？

  127.0.0.0/8, loopback

- ICMP 这个协议最主要的目的为？同时做为『响应』的类别为第几类？

  做为网络检测之用，为第 8 类 (echo request)

- IP 封包表头有个 TTL 的标志，请问该标志的基本说明为何？其数据有何特性？

  为该封包的存活时间，该时间每经过一个 node 都会减少一，当 TTL 为 0 时，该封包会被路由器所丢弃。 该数据最大为 255。

- 在 Linux 当中，如何查询每个 port number 对于服务的对应 (filename)

  /etc/services 档案中有纪录

- 什么是星形联机？优点为何？

  利用一 hub/switch 连结所有的网络设备的一种联机方式，最大的好处是，每个『网络设备与 switch 之间』都是独立的， 所以所以每个主机故障时均不会影响其它主机的联机。

- 请说明 CSMA/CD 的运作原理？

  发送流程

  1. 主机欲使用网络时，会先监听网络，若网络没有被使用时，才会准备传送，否则继续监听；
  2. 当数据传送钟，发现有碰撞情况时，则会重新监听网络，并且重新发送一次该封包；
  3. 若重复发生碰撞 16 次，则网络会瘫痪；

  接收流程

  4. 主机如果没有在传送数据，则会监听网络，并且主动在接收的状态下；
  5. 若接收到一个封包，并且该表头所载 MAC 为本身的网卡卡号，则开始接收该封包，否则将该封包丢弃；
  6. 接收过程当中如果发生封包碰撞，则会通知原发送主机碰撞的数据；
  7. 封包接收完毕后，会以 MAC 表头所载长度同时分析本封包长度，若发生问题，则会通知对方重新传送。

- TCP/IP 这两个通讯协议必须要在一起才能生效，为什么？

  因为 IP 协议仅定位出 IP 的所在处与路由，并没有沟通协调的能力， 至于 TCP 封包则具有目的端、本地端程序之间沟通的能力，但无法直接传送封包。 故 TCP/IP 常会放在一起讲。

参考数据

特别感谢：
本文在 2002/07 发出之后，收到相当多朋友的关心，也从而发现了自己误会的一些基础的网络理论，真的是感谢好朋友 Netman 兄与 ZMAN 兄的指导！这篇短文在 2003/08/03 做了相当大幅度的修订，与原来的文章 (上次更新日期 2002/09) 已经有一定程度的差异了，希望网友们如果有时间的话，能够再次的阅读， 以厘清一些基本概念喔！

- Study Area 之网络基础：http://www.study-area.org/network/network.htm

- Request For Comment (RFC) 技术文件：ftp://nic.merit.edu/internet/documents/rfc
- Request For Comment (RFC) 技术文件：http://www.rfc-editor.org/
- Hub 与 Switch 的迷思：http://www.study-area.org/tips/hub_switch.htm
- BBS 上的问答收集
- Robert Breyer & Sean Riley 着、风信子、张民人译，『Switched & Fast 以太网络』， 旗标出版社
- 粘添寿着，『Internet 网络原理与实务』，旗标出版社。
- Phil Dykstra, Gigabit Ethernet Jumbo Frames：http://sd.wareonearth.com/~phil/jumbo.html
- 台湾地区 IP 核发情况：http://rms.twnic.net.tw/twnic/User/Member/Search/main7.jsp?Order=inet_aton(Startip)
- 全球 IPv4 的统计：http://www.twnic.net.tw/ipstats/ipv4stats.php

局域网络架构简介与 Linux 版本选择

最近更新日期：2006/07/13

在前一章『网络基础』当中， 我们介绍了比较多理论方面的网络相关信息，也大略的介绍了一些简单的网络联机媒体需要注意的事项。 在这一章当中，我们将会继续讨论在一个小型企业或家庭里面的小型局域网络规划， 以让您的所有计算机主机都可以直接以以太网络进行数据的连接啊！ 一般来说，内部局域网络都希望直接使用私有 IP 来设定沟通环境， 直接以简单的星形联机做为网络施工的主要类型，底下就分别来谈一谈如何规划您主机在星形联机所应该要放置的状态， 以及主机应该使用何种版本的 Linux distribution 较佳呢？

局域网络的联机

谈完了网络基础后， 现在就让我们实际的来将家里或者小型企业内部的全部计算机给他连接起来吧！ 当然啦，我们这里主要介绍的是小型局域网络的架构，如果是比较大型的企业内部， 那么将『配线盘、线路设计、墙上网络孔』分别拆开施工的结构化布线会比较妥当， 不过，结构化布线并非本文所想要讨论的，如果您的企业有需求的话，可以向专业人士寻求协助，举例来说，酷学园(http://phorum.study-area.org)的 ZMAN 兄就是一位很棒的网络布线专家。 无论如何，先来将所有的网络硬件联机起来吧！

什么是局域网络

局域网络 (Local Area Network, LAN) 顾名思义就是在我们所属区域的所有网络嘛！ 如果由前一章网络基础提到的相关网域概念来看的话， 我们可以说局域网络就是在同一个网域内所有联机主机及网络媒体的统称。 此外，局域网络通常是在同一个物理网段以内的，虽然藉由网络地址的设定可以规范出许多不同的 IP 网段， 不过，通常我们还是会称这样的环境为局域网络的。

总之，局域网络就是在您主机附近的网络环境，通常是在同一个物理网段内，IP 网段通常也在同一网域内。比如说大学的学生宿舍内的网络环境，其实都是透过整栋大楼的 switch 集中联机， 所以是在同一个物理网段内，就常被称为局域网络啰。

相对于局域网络而言，局域网络外的环境就可以被称为广域网络 (Wide Area Network, WAN)。这也是为什么在您家里的调制解调器上面的网络插孔上，总是有着 LAN 与 WAN 这两个插孔与灯号的存在啊！ ^_^

由前一章我们知道 Internet 其实就是由 INTERNIC 所维护的一个架构松散的联机接口 (其实也不能说是维护，只是 INTERNIC 有提供一些技术文件以及 Public IP 申请的相关信息公布而已)， 任何人只要能够取得 Public IP 就可以连上 Internet 啰，同时 Internet 上头也是没有王法的地方， 要特别留意您的联机啊！OK，那什么是 Intranet ？相对于 Internet 是没有专属维护者的接口而言， Intranet 则是专属的私人网域，只是这个网域使用的是类似 Internet 的联机架构， 例如使用私有 IP 架设 TCP/IP 局域网络的环境就能够被称为是 Intranet 啰。

如果要定义的更为狭义的话，我们可以说，局域网络就是在同一个物理网段的环境内， 使用私有 IP 或者是局域网络适用的通讯协议所串起来的一个网络环境。 既然局域网络常常使用私有 IP ，那么局域网络可否连上 Internet 啊？『当然可以』！ 简单的话，可以使用 IP 分享器来取得 Public IP 上网，复杂的话， 可以使用 Linux 主机架设 NAT (Network Address Translation) 服务器来转址， 就能上网啦！由于私有 IP 不会与 public IP 冲突， 并且也可以避免直接与 Internet 的数据互通，可以减少很多被主动攻击的情况。 所以啊，您的局域网络最好还是设定私有 IP 比较妥当呐！

局域网络的布线规划

记得以前听 ZMAN 大哥某场演讲的时候提到，网络布线是『数十年大计』中最重要的一环， 因为『服务器主机能力不够时换主机就好了，Switch 交换力不足时换 switch 就好了， 但如果布线不良，难道要拆掉房子将管线挖出来重新安装设定？』所以说，最初规划的布线严谨度真的会影响到未来网络的分布情况啊！所以说， 如果您的企业『整栋大楼需要重新布线』时，真的非常建议您务必要找寻专业网络布线专家帮忙设计规划， 因为连一个小小的机柜配线箱都有大学问～设计的好的话，每部独立的主机要改线路、 要换插孔都变的很简单！而且主机到墙上插孔的距离也会变的很短，维护也会很方便！线段也会很美观！ 当然啦，如此一来，网络线材的选择也就不能够用太差的！而且网络布线经过折角区时， 也需要特别留意施工呐。

但是本文讨论的是一些比较小的局域网络环境，这样的环境可以是在一间办公室内而已， 所以我们这里谈到的大多是比较单纯的布线状态，并没有考虑到办公室外部的环境， 所以参考本文时，请特别留意这种差异性喔！

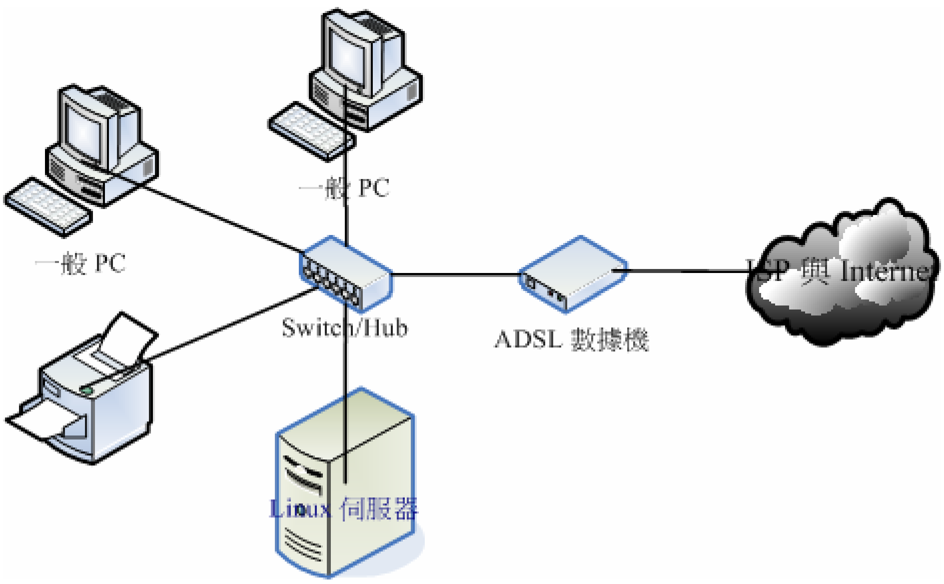
在这样单纯的环境中，我们可以利用一个以 switch 为中心来串连所有设备的星形联机 (star topology) 架构来设计我们的局域网络啊！在这样的环境中您需要担心的是『 那我的 Linux 服务器要放在那个地方？』会考虑 Linux 服务器是因为鸟哥假设你需要在你的局域网络内架设对 Internet 开放网络的服务！而 Linux 是否具有 Public IP 对于主机的维护与设定的复杂度有很大的影响，所以当然需要考虑啰！ 底下鸟哥以目前在台湾挺流行的 ADSL 利用电话线路上网的环境来说明几种联机状态：

Linux 主机直接连到 Internet 的环境：

在底下的环境当中，鸟哥假设我们仅有一条 ADSL 的对外联机，也就是说， 我们的 Linux 与一般 PC (不论何种操作系统) 都是透过同一条线连到 Internet 上面去的。

让 Linux 与一般 PC 在同等地位：

如果您使用的 ADSL 是多 IP 的条件 (例如拨接可以给予 2-8 个 IP 的情况)，那么最简单的方式就是如下图一的联机模式：



图一、Linux 服务器取得 public IP 的联机方式之一(具有多个可用 IP 情况)

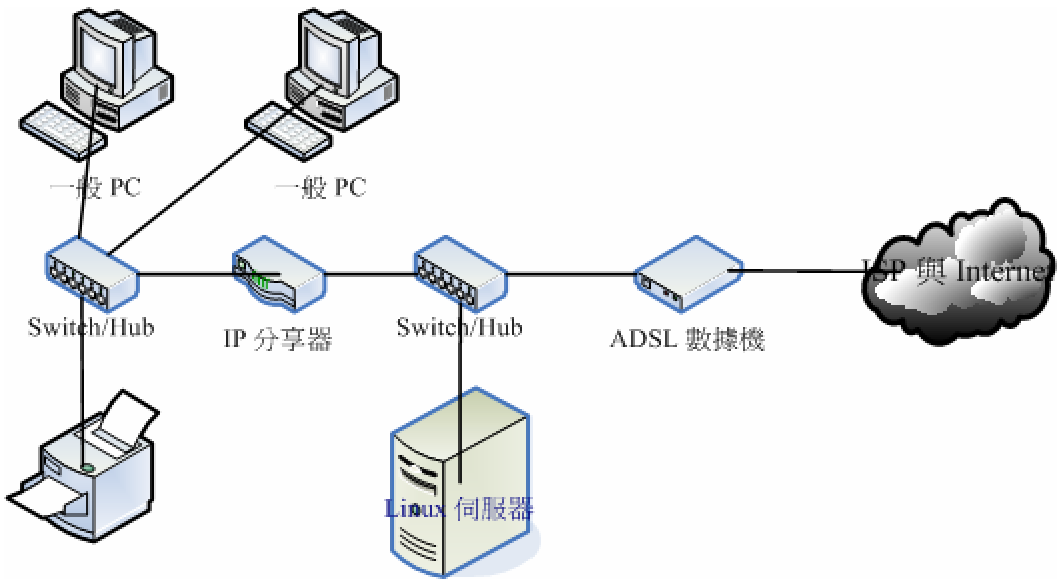
在这种联机模式当中， Linux 与一般 PC 或打印机都是同等地位，并没有谁比较『大尾！』^_^ 如果不急着连上 Internet 时，那么每个设备都给予一个同网域的私有 IP 就可以进行网络联机的工作了， 您也可以很快乐的使用打印机或者是网络上的芳邻等等工作。 此外， Linux 服务器也可以作为内部的档案服务器或者是打印机服务器等等。

当需要连上 Internet 时，每部计算机 (包括 PC 与 Linux 主机) 都可以直接透过拨接连上， 而由于拨接是在每部机器上面『额外增加一个实体的 ppp0 接口』，因此， 拨接上网之后每部主机还是可以使用原有的局域网络内的各项服务，而无须更动原本设定妥当的私有 IP 。这样的情况对于一般家庭使用者来说，可以算是最佳的解决方案啦！因为如果您的 Linux 主机挂点时， 其它个人的 PC 是不会被影响的！

不过这样的环境对于小型企业主来说，却不好管理。因为无法掌握每个员工实际上网的情况， 而且对于防火墙来说，『根本就是一个没有防火墙的环境』，所以， 是没有办法对员工进行任何实际网络的掌控的，并且由于网络内外部 (LAN 与外部环境) 并没有明确的分开，网管人员对于进入 client 端的封包是没有任何管理的能力， 所以对于网络安全来说，是很难管控的一种环境啊！因此对于企业来说，不建议这种环境。

让 Linux 与一般 PC 分开：

如果您有多个可用的 public IP ，并且您的 Linux 服务器主要是提供 Internet 的 WWW 或 mail 服务，而不是作为内部的档案服务器之用，那么将 Linux 服务器与内部的网域分开也是个可行的方法， 而且 Linux 拥有 public IP，在设定与维护上面也不困难，如下所示：

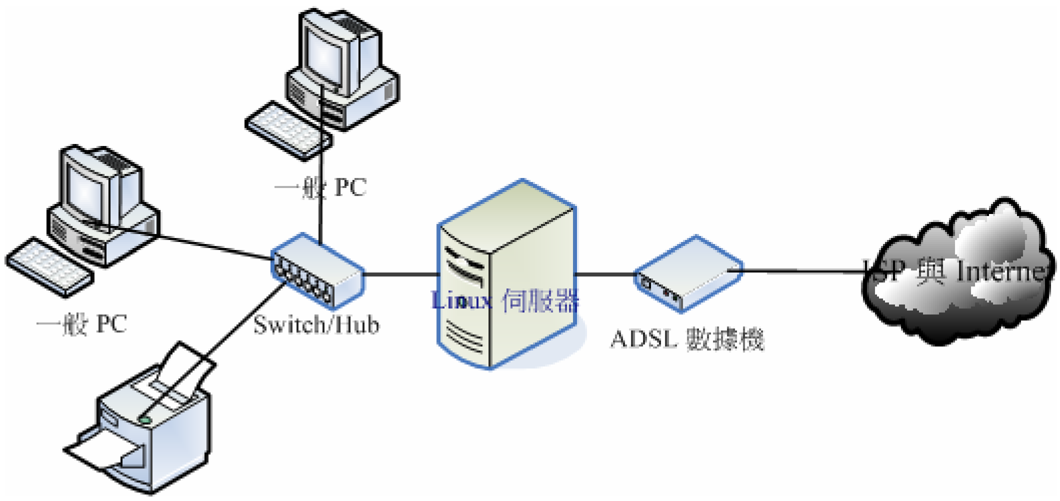


图二、Linux 服务器取得 public IP 的联机方式之二(具有多个可用 IP 情况)

所有的 LAN 内的计算机与相关设备都会在同一个网域内，所以传输速度是没有问题的， 此外，这些计算机要连出至 Internet 时，必须要透过 IP 分享器， 所以您也可以在 IP 分享器上面设定简单的防火墙规则， 如果 IP 分享器可以换更高阶的设备时，那么您就可以在该设备上面架设规则较为完整的防火墙， 对于内部主机有相当程度的管理，并且好维护啊！

让 Linux 直接管理 LAN:

如果您不想要购买 IP 分享器的话，那么直接利用 Linux 服务器来管理就好了啊！ 没错啊！那么你可以这样布线：



图三、让 Linux 管理 LAN 的布线情况

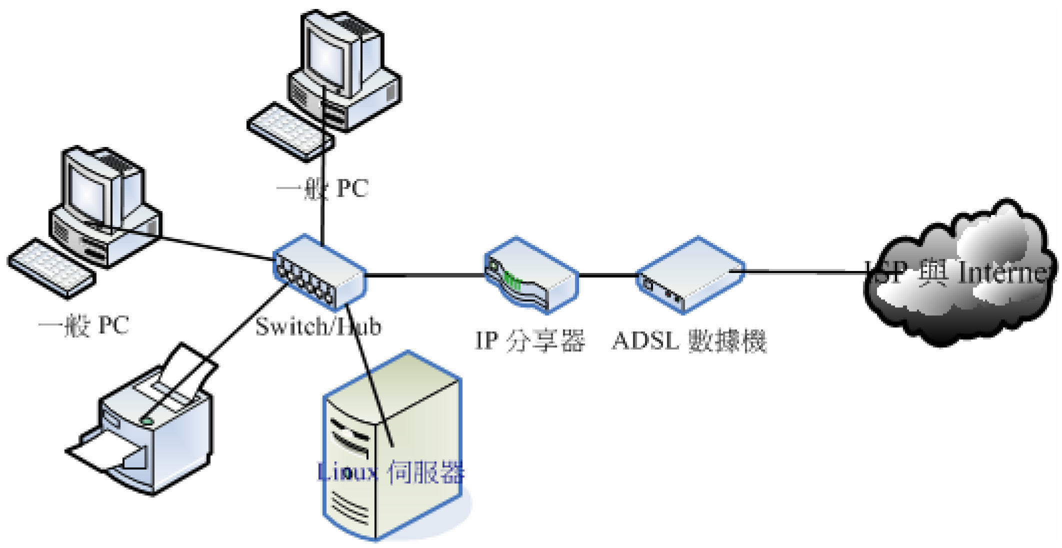
这种情况下，不论你有多少个 IP 都可以适用的，尤其是当你只有一个 IP 时，就非得使用这种方式不可了。 让 Linux 作为 IP 分享器的功能相当的简单，同时 Linux 必须具备两张网络卡，分别是对外与对内，由于 Linux 依旧具有 public IP ，所以在服务器的设定与维护上相当的简单， 同时 Linux 服务器可以做为内部网域对外的防火墙之用，由于 Linux 防火墙的效能挺不错 加上设定也很简单，功能却也是很不错的！因此，网络管理人员也较能进行较完善的掌控， 并且， Linux 服务器也要比高阶的硬件防火墙便

宜多了！ ^_^ 鸟哥个人是比较喜好这种方式的联机啦！

不过，我们都知道『服务器提供的网络服务越单纯越好』， 因为这样一来主机的资源可以完全被某个程序所使用，不会互相影响，而且当主机被攻击时， 也比较能够立即了解是那个环节出了问题。但是如同图三的状况来说的话， 由于内部的 LAN 是需要通过 Linux 才能联机出去，所以 Linux 挂点时，整个对外联机就挂了， 此外， Linux 的服务可能就太复杂了点，可能会造成维护上的困难度。 但对于小型区网来说，图三这种架构还是可以应付的来的啦！

Linux 主机放在 LAN 里面：

瞎密？我们的 Linux 主机放在 LAN 里面？有没有搞错啊？没搞错啊～ 比较大型的企业通常会将他们的服务器主机放置在机房内，主要是在 LAN 的环境下， 再透过防火墙的封包重新导向的功能，将来自 Internet 的封包先经过防火墙后才进入到服务器， 如此一来可在防火墙端就砍掉一堆莫名其妙的侦测与攻击，当然会比较安全啊！ 这种架构还依防火墙的多寡而又可分为非军事区(DMZ)的配置，不过，太麻烦了～不建议初学者直接使用。 底下我们仅介绍较简单的架构来说明：



图四、Linux 主机放在 LAN 里面的布线情况

这里我们以一个较简单的图示来说明，所以利用的还是 IP 分享器，可能的话， 您可以将 IP 分享器换成 Linux 主机来架设防火墙，也是一个不错的选择啊！ 反正现在计算机天天在升级，升级后的旧配备其实就可以作为 Linux 防火墙之用了！ 反正防火墙又不需要什么硬盘与强效的显示或者 CPU， 只要有不错的网络接口就能够达到不错的防火墙效能了。

不过这里得再次的强调，Linux 服务器主机若放在 LAN 里面，则当你要对 Internet 提供网络服务时，防火墙的规则将变的相当复杂，因为需要进行封包转递的任务，在某些比较麻烦的协议当中， 可能会造成设定方面的困扰。所以，在您初接触 Linux 服务器时， 不建议新手使用这种联机架构，避免由于失去信心而没有动力学习～(@_@)。

每种联机的方式都有其适用的使用者群，所以没有那个是比较好的， 完全是看您自己的网络环境而定喔！了解了简单的网络布线方式后， 底下我们来谈一谈几个常见的网络组件的名词吧！

- Node (节点):
  连接在网络上的， 具有网络卡卡号的设备都可以是节点。 包括服务器、工作站、路由器与网络打印机等等，都是一个节点； 如同网络基础内谈到的， IP 封包内会有个 TTL 的封包存活时间，该时间与封包经过的节点数有关呐!

- Client (客户端):
  向主机端主动发出联机要求的就称为 Client 。

- Server (服务器端):
  在网络上面提供网络相关服务的， 可以响应客户端的联机要求者，就可以称为是服务器了。不过 client/Server 的架构当中， 每部主机的身份可能随时会改变啊!举例来说，当 mail server 要传送数据时， 必须要向 DNS 主机询问目标 email 的 IP 地址，此时 mail server 反而是 client 端呐!

- Interface (网络接口):
  网络接口除了实体网络卡之外， 透过点对点 (point to point) 联机方式所建立的 PPP0 接口也是一个实际的界面。 此外，每张网络卡上面都可以设定多个 IP ，那些多的 IP 则设定在虚拟的网络接口上!

- Network Interface Card, NIC(网络卡): 可以说是最重要的网络组件了! 因为所有想要连上以太网络的主机都需要有以太网络卡!

- Workstation (工作站):
  没有对 Internet 提供网络服务， 但是提供使用者登入进行学术研究，例如数值模式仿真、大型程序编译与开发等等的高阶主机， 都可以被称为工作站。

OK!我们现在知道要连上以太网络组成的局域网络，就得要有网络卡、网络线、网络集中媒体(hub/switch)、连上 Internet 的调制解调器等等，在这里鸟哥将防火墙、路由器等等设备归类为主机， 因为基本上，这些组件内部一定会含有一个网络卡，只是操作系统的精简程度与软件功能的不同就是了。 那么这些所需要的网络硬件又该如何挑选呢?

网络媒体选购建议

在开始底下的介绍之前，您必须要对于跳线、并行线、RJ-45 网络线、Hub/Switch 的优劣等等有一定程度的了解，请再前往网络基础看一看。 此外，在非我们局域网络内的设备，例如调制解调器，那就得向您的 ISP 询问了! 一般来说， 调制解调器是中华电信提供给用户的，然而由于『中华电信因为不同批次安装的调制解调器模块不同， 所以会有不一样的连接与线材处理方式!』例如早期的调制解调器 (有的朋友会称 ADSL 调制解调器为小乌龟) 连接到计算机的网络卡是使用跳线，但是近期的调制解调器却使用的是一般的并行线! 所以请特别向您的 ISP 询问才行。底下主要针对局域网络内的网络媒体来进行介绍与说明。

网络卡:
目前常见的 ADSL 拨接与固定制，以及 Cable Modem 的联机上网方式都需要藉由以太网络卡的辅助， 我们也知道以太网络卡的规格实在太多了，所以底下我们就简单的来作个说明吧!

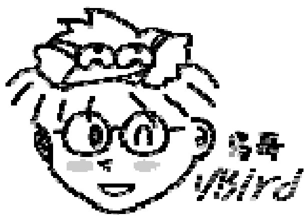
一般来说，目前的新主机几乎都是内建 gigabit 的以太网络卡了，所以您不需要额外购买网络卡。 不过，

这种内建的网络卡通常芯片的启动驱动程序比较麻烦，您必须要先参考他官方网站所提供的驱动程序安装手册后， 才能够顺利的驱动哪！如果想要额外购买网络卡的话，以现在的网络速度与联机质量来看， 一般家庭使用 10/100 Mbps 应该是足够了，不过如果有特殊需求的话，买个 10/100/1000 Mbps 的超高速以太网络卡也不错喔！

如果是想要作为 Linux 服务器的话，那么您的网络卡可能必须要购买好一点的。 举例来说，某些主机板内建便宜的 gigabit 网络接口，但越便宜的网络接口可能会造成损耗较多的 CPU 资源， 如果能够购买类似 Intel/3Com 等知名品牌的 gigabit 适配卡，不但传输较为稳定， 并且可以降低系统资源的耗费，是有一定程度的帮助的。另外，如果强调高速的话， 甚至可以选用 PCI-Express 的网络卡，而不使用传统的 PCI 接口。因为 PCI-Express 的传输频宽更高。

不过，如果是一般家用，或者是准备用来作为学习机之用的主机，那么万一网络卡芯片无法驱动时， 请先买个螃蟹卡 (芯片是 Real Tek 8139) 来作为练习之用， 因为 Linux 本身就支持 Real Tek 8139 的芯片，您不需要额外的驱动程序，这样会方便学习啊！ 而且该网络卡也很便宜(大卖场一片不到 300 块台币)。

Tips:
如果要玩 Linux 又想比较顺畅的玩弄 Linux 时，请不要坚持使用 Linux 捉不到的网络卡！否则那份失望的心情 ......会让您失去很多很多的耐性与信心啊～



螃蟹卡最好认的地方在于其芯片上面有个类似螃蟹的 Logo ，以前鸟哥曾经在大卖场上面逛大街时， 还『踢飞』过一整排螃蟹卡～便宜到都放在地上而已～ @_@

Switch/Hub：
就如同网络基础里面曾经谈到的， Hub 是共享媒体而 Switch 是具有独立频宽的非共享媒体。因此以效能以及速度来看，当然是 switch 比较好用啊！ 不过，如果您是一般家庭用户，只是要作简单的上网等工作，是没有必要购买太好的集线器的， 建议使用一般大卖场可以买到的 5 port 的集线器即可 (差不多 500 块台币的就不错了)。

不过如果你常常在区网内传送大量的数据，例如一次传输就得要传送 GBytes 的数据时， 那么网络的整体速度需要很详细的考虑喔！包括网络卡最好使用 gigabit ， 当然中间的联机设备最好买支持到 gigabit 速度的 switch 啦！因为 10/100/1000Mbps 的 switch 要比 10/100Mbps 的设备快上十倍，速度可是差很多的啊！如果您的设备还需要更快时， 例如鸟哥之前服务的实验室内部的 cluster (丛集式计算机群)，则购买的 switch 甚至需要支持 Jumbo frame 这种支持大讯框的硬件架构才行，否则速度上不来啊！

购买 switch/hub 时，注意到该硬件是否具有 N-Way (自动协调速度机制) 以及 Auto MDI/MDI-x (自动跳线机制)， 这样可以不需要考虑手动切换速度以及网络线购置错误的问题。不过，以目前的消费水平来说，建议添购时，直接购买 gigabit 的 switch 吧，扩充性会比较好呐！

网络线：
在所有串连网络的设备当中，网络线是最重要，但是却也最容易被忽略～ 除了网络线的等级会影响到联机速度外，网络线所在处是否容易被压折？ 是否容易有讯号衰减？自己压制的 RJ-45 接头是否通过测试？ 网络线是否缠绕情况严重？ 都会影响到网络的传输优劣！所以，虽然我们常常讲要确认主机与 Switch 是否有连接成功可以看 switch 上的灯号，但是很多时候虽然灯号是亮的，不过由于网络线折损严重的问题，也会导致联机质量不良喔！

一般来说，『个体户』与小型企业通常网络线是直接放在外部的，这种情况您发现网络怪怪的时， 可以直接更换线路，不过，如果是如同中大型企业将网络线直接埋在墙内， 或者是在管线当中，发现问题时，真的很麻烦～所以才需要专业人才的辅助啊！

Tips:
一般来说，越高等级的网络线，最好不要自行制作，因为一个小小的 RJ-45 接头的压制， 由于蕊线裸露程度的不同，就会影响到电子屏蔽效应的优劣了。Cat 5 等级的线材还可以自行压制， 比他还高等级的，最好还是买现成的吧！ ^_^



无线网络相关设备：
现在的网络环境除了传统的以太网络之外，其实还有一个也是很常见的喔，那就是无线网络啦！ 无线网络会流行主要的原因除了笔记型计算机能力越来越强，使得很多朋友直接以笔记型计算机取代桌上型计算机之外， 无线网络的速度目前已经可以达到 54Mbps 那么快了，对于一般只是上网看新闻与聊天的上班族来说， 这样的速度实在是非常快了 (一般的 ADSL 仅是 2M/256K bps 而已)， 所以要买无线网络设备 (含基地台与在 client 端的无线网卡) 来做成局域网络，其实也是可以啦!而且还可以省去网络线的施工呢!

不过，无线网络最大的问题常常在于『无线的安全性』方面，因为是无线的设备， 所以『基地台如果没有做好防备措施的话，常常会导致 LAN 内的主机数据被窃取』， 这可是非常大的问题喔！可千万不要小看这个问题，吃上官司常常是由于忘记网络安全啊！ 记得购买无线网络基地台时，注意他可否『限制 MAC 』，如此一来，至少可以所网卡， 只让指定的网卡可以使用您的无线基地台，比较安全啦！

关于其它配件：
事实上，整个网络环境可不止上头提到的这些咚咚而已，还包括硬件防火墙、路由器、桥接器等等的， 当然，这些设备贵的话也有上百万的，但您的环境是否需要用到这么好的设备，那就见仁见智啊～此外， 为了环境的美观与生活的便利，您总不希望走在路上被网络线所绊倒，也不希望因为网络线绊倒你导致网络媒体掉落， 结果........损失了一堆 $$ 吧～所以啰，在网络线的转角处必须特别注意线材的保护， 在平面地上则需要特别使用压条给予固定，在牵线施工的时候尽量让线材沿着墙角或者是墙面上的既有物品，如此则除了保持工作场所的美观之外，还能够增加工作场所的安全性啊！ ^_^

此外，『计算机上网的速度并非完全取决于网络频宽』举例来说， 玩在线游戏时，大家都以为网络频宽需要很高规格，其实....根本不需要！因为 3D 联机游戏最主要的速度瓶颈应该是在于『3D 显示』而不是网络。这是因为网络仅传送一些数据给您的主机， 而您的主机再在自己的硬盘里面将图形取出，并且使用 3D 绘图卡将画面绘制到您的屏幕上。 所以，显示速度或者是 CPU 不够力时，才会发生联机游戏的顿点。否则就是联机游戏服务器本身的负载 (loading) 太大，导致主机响应有较多延迟，就产生 lag(顿点) 的问题啦！

另外，包括您主机使用的数据是否具有快速的传输接口也有关。举例来说，如果您的主机使用 USB 1.1 (最大传输 12Mbps)，但网络速度可达 10/100/1000Mbps ，那当您要在远程使用这部计算机的 USB 装置内的数据时，最大速度会是『 12Mbps 』，也就是最慢的那一个组件。所以啊， 网络速度慢的时候， 不要以为只要增加网络频宽就好了，要确切的找出问题啊！ ^_^

事实上，选购网络媒体所需要考虑的参数实在太多了，并且没有一定的依据， 完全与使用者的使用环境与未来功能性有关。不过，如果着眼在单纯的硬件速度上面的话， 那么选购时考虑『我的网络速度可接受的

最低速度为何？』去考虑吧！ 如果行有余力的话，再来考虑『我的环境需要多稳定的设备来达成？』 其它的，那就得要靠您自己摸索啰！ ^_^

内部联机的网络参数与通讯协议

除非您已经具有相当熟练的 Linux 系统与服务器架设维护经验， 否则不建议您使用上面图四所介绍的联机模式， 对于初接触 Linux 服务器架设与维护的朋友来说， 将您的联机模式设定成图三应该是个不错的选择， 除了可以让您简单的就将服务器架设成功之外，也可以让您以 Linux 做为内部 LAN 的防火墙管理中心， 对于未来的学习成长方面较有帮助啊！ ^_^

在图三的环境下， Linux 必须要具有 router(路由器) 的能力，所以当然必须就要有两个接口， 一个接口与 Internet 沟通，另一个接口则与内部的 LAN 沟通。 那么为什么鸟哥说的是『两个网络接口』而不是 『两张网络卡』呢？原因很简单， 因为一张网络卡可以设定多个 IP 啊！因此，在 Linux 当中一张网络卡可以具有一个以上的 IP 呢！由于一个 IP 即为一个网络接口，因此只要两个网络接口 (不论有几张网络卡) 即可进行 NAT 的设定啦！所以自然一个网络卡即可啰！不过，鸟哥个人还是比较喜欢并且建议两张网络卡的啦， 将内外网络环境完整的分开，让您的内部网络效能较佳一点！

关于与 Internet 的联机方面，目前在台湾最常见的有电话专线的 ADSL 联机模式、 利用电视缆线搭设单向或双向的 Cable Modem ，以及例如学术网络的固定 IP 的专线等等。 这些联机的方式我们将在后续章节继续介绍的。至于内部的 LAN 我们则建议使用 Private IP 来设定喔！ 鸟哥通常喜欢使用 192.168.1.0/24 这个 Class C 的网域，没什么特殊原因，只是因为....我喜欢！ ^_^ 在选定了 Private IP 的网段后，您必须要有『IP, Network, Netmask, Broadcast, Default gateway 以及 DNS 服务器的 IP』等等的设定值。 假设我 Linux 主机的对内 IP 为 192.168.1.2 ，则 在图三内的 LAN 内的 PC 之网络相关设定参数则为：

- IP:设定为 192.168.1.1~192.168.1.254 ，但 IP 不可重复;
- Netmask：255.255.255.0
- Network：192.168.1.0、Broadcast：192.168.1.255
- Default Gateway：192.168.1.2 (路由器的 IP)
- DNS：暂时使用 168.95.1.1

安装什么通讯协议

什么是通讯协议呢？简单的说，通讯协议就是一些标准与规则。一个社会要能够正常的运作， 必须要有一套标准与公正的游戏规则，大家都遵循这个规则，那么这个社会才能够没有问题的营运下去， 所以我们才会有警察、司法、检调单位等等来维护我们的公正性。同样的，在网络的社会中， 要让数据能够透过网络媒体传送，那么硬件制造商、软件开发商，就必须要共同遵循同一套标准， 这样整个网络社会才能够正常无误的进行数据的传递。

目前网络社会最通用的通讯协议就是 TCP/IP 了！TCP/IP 订定了 IP 基础与路由协议等信息，让我们的网络世界可以互通有无，此外，在局域网络内部， 由于是小型的网域，事实上还可以透过简单的通讯协议来达到数据传输的目的，例如 NetBEUI 就是一个常见的简易通讯协议。 在 Linux 系统当中，只要将网络参数设定妥当，那么 TCP/IP 就已经被启用了，所以您不需要额外的再安装其它的通讯协议。不过，如果您需要将你的 Linux 系统中的硬盘空间分享给同网域的 Windows PC 时，那么就需要额外的加装 SAMBA 这

个服务器软件才行。相关的 SAMBA 数据我们会在后面的章节提及。反正不管怎么说，目前 Internet 就是经由 TCP/IP 来进行连接的，而 Linux 本身就支持了 TCP/IP ，所以不需要额外的安装有的没的说！

至于在 Windows 部分就比较麻烦一点，因为在较大型的企业当中，还需要额外的考虑到 Windows Server 所提供的服务，那么在 Windows Clients 端就得要相应的启动某些通讯协议才行。一般来说，在 Windows Client 系统里面，最常见的两个通讯协议就是 TCP/IP 以及 NetBEUI 这两个通讯协议了。如果您只想让 Windows 与 Linux 能够藉由网络上的芳邻互通有无，那么启动 TCP/IP 也就够了 (因为 SAMBA 是藉由 NetBIOS over TCP/IP 来达成数据传输的) ，不过，也可以同时启动 NetBEUI 这个通讯协议就是了。相关的说明可以参考 网络基础 以及后续的 SAMBA 服务器等章节。

Linux distributions 的选择

在『鸟哥的 Linux 私房菜 -- 基础学习篇』当中我们提到过 Linux distributions 可是多的跟什么一样，那么你该如何选择你的 Linux distributions ？基本上，哪一套 Linux distribution 鸟哥觉得都差不多，但是每一套 Linux distributions 当初推出的预设使用群与他的理念可能不太相同， 就导致不一样的 distributions 的用途差异了。并且由于您所需要的软件以及新添购硬件的支持度等等， 也会让您的 distributions 挑选有些难度喔！

主机硬件的选择

主机的功能与硬件的选择可以是息息相关的。怎么说呢？ 因为老旧的硬件无法提供高效能的环境，所以对于需要强效的 Proxy, WWW (MySQL) 等服务器， 老旧的主机可能就无法提供有效的环境了。不过，如果是不需要很好性能的防火墙、 DHCP 主机与 NAT 服务器时，那么目前被淘汰掉的 P-III 等级的主机就非常足够了！ 像卧龙小三大师就曾使用赛阳 400 等级的 CPU 做成流量不大的 WWW 主机， 所以，旧式主机可不要随便丢弃啊！

如果是像防火墙、NAT 主机、DHCP 主机等等，那么一般的 P-166 等级左右的主机系统， 配合一张 10/100 Mbps 的网络卡就很不错了！不过如果 LAN 与 WAN 的流量太大的话， 那么最好考虑 P-III 等级的主机，并且配合一张更好的网络卡，这样在效能上应该会比较好。

如果像是 WWW, Proxy, mail server 等需要常常读取硬盘数据的服务器， 则除了主机系统最好要到 P-III 以上等级，网络卡不能太烂之外，硬盘的效能最好也必须要好一些。 而且 partition 的情况也必须要特别注意，最好将常常被存取的档案数据独立到一颗硬盘中， 可能的话，甚至可以组成 RAID (磁盘阵列) 来负责数据存取，让系统效能可以不被某些组件限制。 此外，内存的容量也需要特别留意，如果太小的话，可能会造成系统的效能不彰。 反正如果要架设好一点的主机，那就得要各个组件都加以考虑喔！

省不省电有所谓

前两年出产的处理器与相关的个人计算机零件，很多都非常的『耗电』， 您或许觉得『耗电』又没有什么了不起，一部冷气机的耗电量都要比计算机来的大！ 是没错啦，不过，如果仔细想一想若一部用来作为单纯的 IP 分享器的个人计算机主机， 偏偏他的耗电量高达 200W 的话，以一度电 2.4 元新台币来计算，假设这部主机是全年无修的， 一年可得花费 4200 左右的新台币啊！如果以 IP 分享器来说，鸟哥是没有计算过 IP 分享器的耗电量， 不过，应该不会超过 50W 吧？如此计算一下，相差 4 倍！您觉得如何呢？

对于一般家庭只有一部主机的情况来说，这问题不大啦！一年仅消耗 4200 左右，还可以接受。 但对于企业主来说，如果他需要 50-100 部主机呢？这可不是不可能的是吧！所以计算一下， 哗！一年要花 21 至 42 万新台币啊！而且这还不包括机房的空调冷却系统。现在看一看，觉得害怕了吧！ 所以说，选购硬件时，除了要考虑机器的性能以及功能需求之外，耗电量能省则省吧！ ^_^

虚拟化技术

现在的主机动不动就给他『双核心』，也就是说在一部主机上面的一颗 CPU 上内建两个实际的运算单元，简单的说，双核心就是表示您的主机上面有两颗 CPU 啦！那我们知道简单的服务其实并不会太消耗系统资源的， 所以，绝大部分的时间 CPU 都是在休眠的状态。那我们也知道最好每部主机仅负责一个网络服务，环境与除错上会比较单纯，对吧！综合这么多的因素，您会想，如果我要安装五种网络服务， 就得要五部主机，而这五部主机却偏偏都在『休眠』的状态，咦！好像很浪费喔！

这就是后来虚拟化技术发展的因素之一啦！所谓的虚拟化，其实就是在你一部实体主机里面， 安装多个操作系统，而且这些操作系统都是同时存在，且不互相干扰的。 举例来说， VMWare (http://www.vmware.com/) 就是一个很知名的虚拟化主机的软件，他可以在 Windows 操作系统里面开启另一个操作系统， 也可在 Linux 里面开启另一个操作系统。这可不是『多重操作系统』喔！因为多重操作系统指的是『 你的主机同时安装多个操作系统在不同的 partition ， 而开机时可以选择不同的操作系统操作』的意思， 与虚拟化『可以在一部主机上面同时开启多个可独立运作的操作系统 』是不一样的！

既然您的 Linux 主机是多核心系统，表示 CPU 资源比较多，可以负荷的工作比较多， 加上目前有的虚拟技术，所以您可以在一部主机上面安装多个操作系统， 这些操作系统可以独立运作，而且，虚拟主机的每个操作系统与看起来就与单一主机没有两样， 呵呵呵呵！这么一来，不但不需要太多主机就能够达到您的多网络服务要求， 而且管理上也没有任何的不方便，当然，因为主机少了，电费的节省也是当然的啦！您说是吧！ ^_^

在 Linux 上面使用的虚拟化技术有很多，不过目前比较受瞩目的当属 Xen 这个玩意儿了， 据说很多预计要推出的新版 Linux distributions 已经预设要内建 Xen 的功能了， 嘿嘿！看来，以后大家可以用一部主机建立多个可同步运作的系统啦！^_^

- Xen 网站：http://www.xensource.com/

---

distributions

需要特别留意的是，旧的 distribution 是不支持很多新的硬件，包括 PCI-Express, SATA 以及很多新的网络芯片。所以，如果您的主机是新的，那么不要想用旧式的 distribution 来安装， 否则，真麻烦喔～因为您必须要自行加上很多的驱动程序！真麻烦～

另外，您可以将目前的 distribution 分成两大类，一类是多功能新鲜货，例如 Fedora ， 一种是强调性能稳定但软件功能较旧的企业用途货，包括 RHEL, CentOS, SuSE 及 B2D 等！ B2D 是台南县网中心卧龙小三大师所提倡的一个计划，最初的构想是想建立一个台湾的中文化 Linux distributions，并且不只是中文化而已，还希望能够提供全国中小学教师快速且安全的建立网络环境的构想下， 所发展的一个优秀的计划!目前这个计划发展的很不错的啊！很着重我们台湾使用者的想法喔!^_^。只是 B2D 使用的是 Debian 这个 distribution，在软件安装上面与一般常见的 RPM 不太相同，好在目前 B2D也有使用 APT 来进行网络安装与升级，可以解决软件安装方面的困扰。

一般来说，我们会建议您如果想要架设服务器时，尽量选择『稳定性较高的企业版』较佳， 因为功能新且强的版本例如 Fedora 由于太强调新鲜货，所以核心与软件的变动情况较为频繁， 那就很容易造成一些困扰，因为很多使用者自行安装的软件可能无法在新的核心上面跑， 所以，只要核心一升级，哇！很多需要编译的软件就都需要再重新编译过， 那么对强调『永续服务的网络环境』来说，不就会造成不良的影响了吗？您说是吧！

由于鸟哥用惯了 RPM 以及 Red Hat 系统的关系，所以在这里推荐您使用 RHEL/CentOS/SuSE 这几个 Linux distributions，因为他够稳定，而且设定上面不难， 而且有新的 distribution 推出，因此适合新硬件架构， 不过，里面的软件版本可能就不会是最新的，这点您可能就得要自行设法啰！ 比较特别的是 CentOS，他不但标榜完全相同于 RHEL ，并且可以直接透过 yum 这个软件进行完整版本的升级，例如由 CentOS 4.2 升级到 CentOS 4.3 只要下达一个指令即可， 既不会影响到原有的设定，升级时所花费的时间又短，所以， 目前鸟哥都是以这个版本来进行服务器的架设啊！

在后面的许多文章说明中，鸟哥将主要以 CentOS 这个版本来进行说明， 此外，也会加入 SuSE 的用法，不过，基本上， CentOS 既然是 Red Hat 系统， 所以后续的说明在 RHEL/Fedora/CentOS 其实都适用啦！您别担心啊！ ^_^

Windows 个人计算机网络设定范例

我们这本书谈论的是以 Linux 主机提供的服务器为主，所以关于 LAN 里面的 Windows 我都将他假设为 Client ，并且不提供网络服务，所以都先以固定的 Private IP 来设定 Windows 操作系统，如果您的 LAN 有其它的考虑，那么底下的设定就看看就好 ^_^。

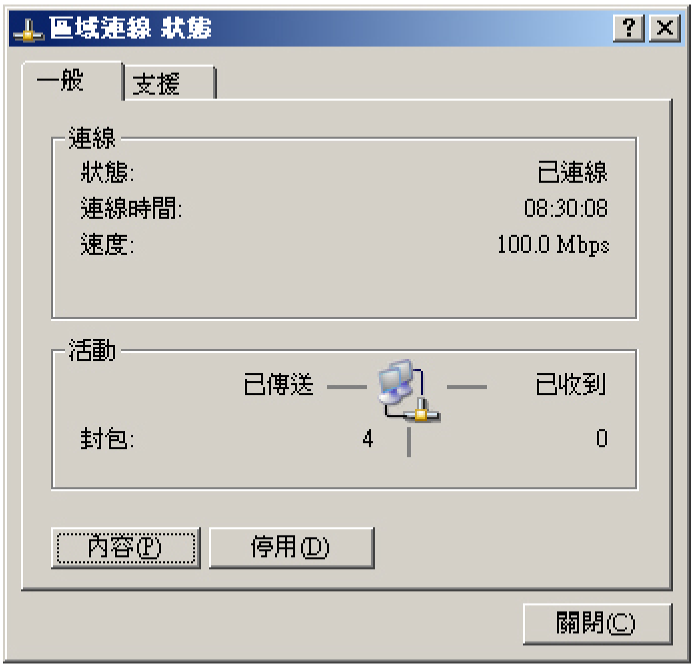
我们在 Windows 系统上所需要的网络参数除了 IP, netmask, DNS 之外，还需要『工作群组, workgroup』与 『计算机名称, Netbios name』等等的设定，此外，我们也可以加上 LAN 里面很常见的 NetBIOS (NetBEUI) 这个通讯协议呐。因此，除非你确定您的网域内还有其它的工作站， 否则『请只要安装 TCP/IP 以及 NetBEUI 这两个协议就好了！』 安装太多反而会有问题呢！底下我们假设你的网络卡都安装好了，并且直接以 Windows XP 这个操作系统来介绍。 没办法，因为 Windows 2000 与 windows 98 以前的版本都已经不太被 Microsoft 支持了， 所以也只好以这个 XP 版本来说明啰。

与网络有关的设定参数：

- IP: 192.168.1.13
- Netmask: 255.255.255.0
- DNS: 168.95.1.1
- Gateway: 192.168.1.2
- 工作群组： birdhouse
- 计算机名称： bird3

详细的设定流程：

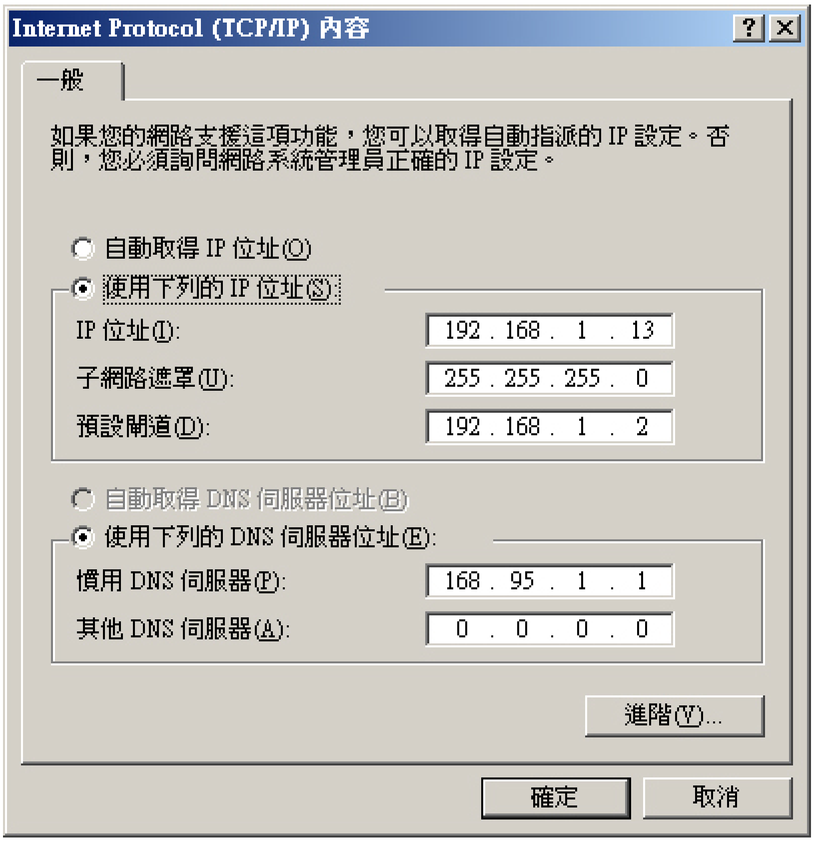
1. 先到『开始』==> 『设定』==> 『控制台』==> 『网络联机』==> 选择『区域联机』该项后，会出现如下图示：


區域連線 狀態
一般
支援
連線
狀態:
已連線
連線時間:
08:30:08
速度:
100.0 Mbps
活動
已傳送
已收到
封包:
4
0
內容(P)
停用(D)
關閉(C)

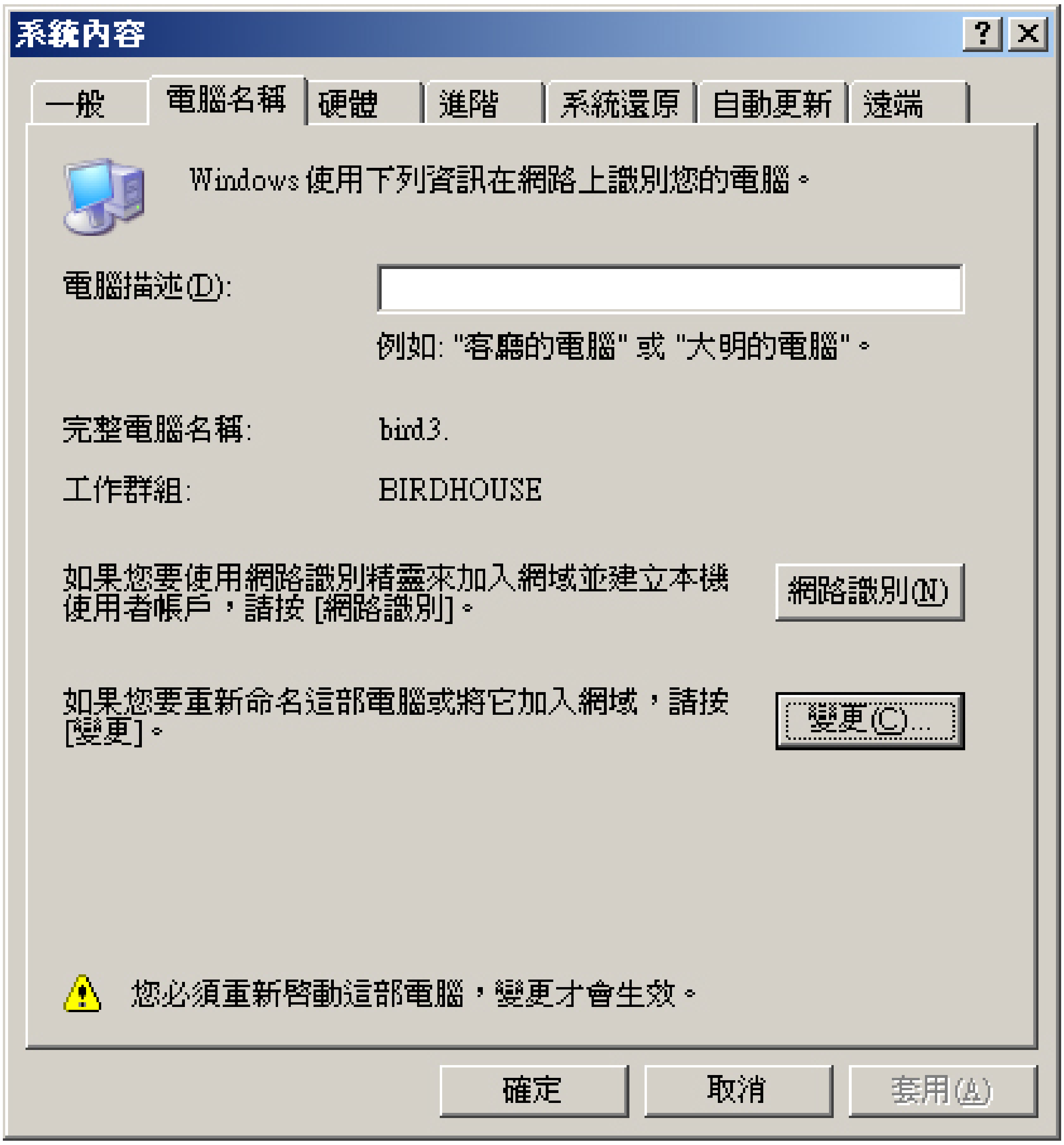
2. 上面画面当中选择『内容』进入设定画面中：



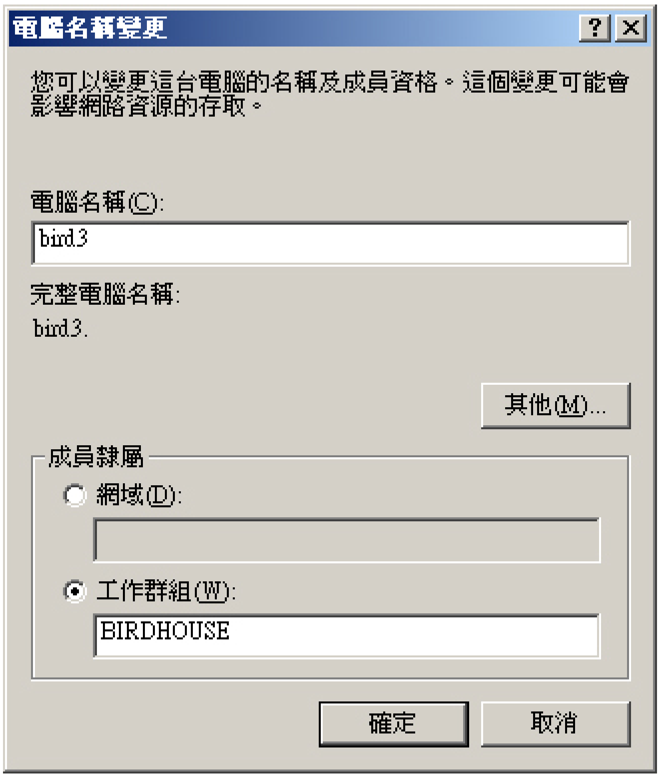
3. 接下来，选择『联机后，将图标显示在通知区域内』， 并且双击『Internet Protocol (TCP/IP)』该项目：



4. 上面的设定不需要再谈了吧？！请填入您的网络参数，并且按下确定即可！ 好了，设定完成之后，再来需要编辑您的网络识别喔！选择 『开始』==> 『控制台』==> 双击『系统』之后出现下图：



5. 在上面的图标当中，如果网络识别是不对的，那么就可以按下 『变更』来重新输入，如下所示：



输入正确之后，只要重新开机，那么就可以使用局域网络啰！

基本上，Windows 的网络参数设定是相当的简单的!鸟哥这里仅介绍修改 IP 与相关网络参数的方式而已。未来如果还需要搭配 DHCP 主机、NAT 主机等等服务器的设定时，会再次的提醒使用者 Windows 的设定信息喔！尤其是 SAMBA 主机的设定中， Windows 的网络识别就显的相当的重要呢！

连上 Internet

最近更新日期：2006/07/24

呼呼！终于要来到修改 Linux 网络参数的章节了！在前面的 网络基础 章节内我们知道了主机要连上 Internet 需要一些正确的网络参数设定，这些设定在 Windows 系统上面的修改则在 局域网络的架构 里面进行了说明。在这一章当中，我们则主要以固定 IP 的设定方式来修改 Linux 的网络参数，同时，也会介绍如何使用 ADSL 的拨接方式来上网，此外，因为 Cable modem 使用者也不在少数，所以我们也说明一下 Cable modem 在 Linux 下的设定方式喔！ 最后，由于笔记型计算机使用者大增，由于笔记型计算机常使用无线网络，因此本文也加入了无线网络的联机介绍啊！

Linux 连上 Internet 前的注意事项：

要连上 Internet 时，由前几章的数据来说，我们知道需要设定一组网络参数， 包括 IP, netmask, network, broadcast, gateway 以及 DNS 主机的 IP 等等， 那我们也知道，其实整个主机最重要的设定，就是『先要驱动网络卡』，否则主机连网络卡都捉不到时， 怎么设定 IP 参数都是没有用的，您说是吧！所以底下我们就来谈一谈， 您要如何确定网络卡已经被捉到，而 Linux 主机的网络参数又该如何设定？

Linux 的网络卡

在 Linux 里面的各项装置几乎都是以文件名称来取代的，例如 /dev/hda 代表 IDE1 接口的第一个 master 硬盘等等。 不过，网络卡的代号 (Network Interface Card, NIC) 却是以模块对应装置名称来代替的， 而预设的网络卡代号为 eth0 ，第二张网络卡则为 eth1 ，以此类推。

我们知道网络卡其实是硬件，所以当然需要核心支持才能驱动他。一般来说，目前新版的 Linux distributions 预设可以支持的网络卡芯片组数量已经很完备了， 包括大厂的 3COM, Intel 以及初阶的 RealTek, D-Link 等网络卡芯片都已经被支持， 所以使用者可以很轻易的设定好他们的网络卡。不过，万一您的网络卡芯片组开发商不愿意释出开放源 (Open Source) 的硬件驱动程序，或者是该网络卡太新了，使得 Linux 核心来不及支持时， 那么您就得要透过：

- 重新编译核心
- 编译网络卡的核心模块

好让核心可以支持网络卡这块硬件啦！但是，重编核心或编译网络卡核心模块都不是简单的工作， 而且有时原始码又可能无法在每部主机上面编译成功，所以万一您的网络卡真的不被预设的 Linux 网络芯片所支持，那么鸟哥真的建议直接换一块被 Linux 支持的网络卡吧，例如很便宜的螃蟹卡！ 免得花了太多时间在硬件确认上面，划不来的！ ^_^

如果您是照鸟哥推荐的，使用 RLT 8139 芯片 (RealTek 8139) 的网络卡 (就是螃蟹卡) ，那您应该在安装 Linux 的时候就已经捉到网络卡了，那真是恭喜您啦！ 因为您的网络卡应该已经可以正常的工作啰！ 那如果在安装的时候并没有捉到网络卡呢？ 该如何是好？那也不用担心，因为您也可以事后才安装网络卡的驱动程序呀！不过，因为 Linux 并不像 Windows 是那样的随插即用，所以需要动一些手术的！

另外，其实有的时候 Linux 的预设网络卡模块可能无法完全 100% 的发挥网络卡的功能的， 所以，有的时候您还是得必须要自行编译网络卡的模块才行喔！当然， 那个网络卡的模块就得要自行由网络卡开发商的官方网站下载了！不过，如果您的网络卡是自行编译安装的， 那么每次重新安装其它版本的核心时，您都必须要自行重新手动编译过该模块。 因为模块与核心是有相关性的啊！

好了，假设您的网络卡已经在主机上面，不论是内建的还是自行安插到 PCI 或 PCI-x 或 PCI-E 的接口上，那么如何确认该网络卡有被核心捉到呢？很简单啊！就利用 dmesg 来查阅即可：

```
[root@linux ~]# dmesg | grep -in eth
117:divert: not allocating divert_blk for non-ethernet device lo
171:divert: allocating divert_blk for eth0
227:divert: not allocating divert_blk for non-ethernet device sit0
228:eth0: no IPv6 routers present

[root@linux ~]# dmesg | cat -n | less
#...前面省略...
   169  3c59x: Donald Becker and others. www.scyld.com/network/vortex.html
   170  0000:00:08.0: 3Com PCI 3c905C Tornado at 0xe800. Vers LK1.1.19
   171  divert: allocating divert_blk for eth0
#...后面省略...
```

透过这个 dmesg 可以发现系统在开机时确实有捉到网络卡，然后透过搜寻行号，就能够找到该网络卡的驱动模块； 当然，您也可以透过 lspci 来查阅网络卡的相关模块啲！ 另外，您也可以透过这个 dmesg 了解到该张网络卡的代号喔！举例来说，鸟哥上面这个讯息就显示： 我的这张网络卡代号是 eth0

```
[root@linux ~]# lspci
```

```
00:08.0 Ethernet controller: 3Com Corporation 3c905C-TX/TX-M [Tornado] (rev 74)
```

看到了吧？鸟哥的某一部主机就是使用 3Com 的网络卡啊！Linux 系统本身就内建了 3c905C 这个驱动程序，所以鸟哥根本不需要自行安装驱动程序呐！真是高兴！ ^_^ 那么这个 3c905 的驱动程序放在哪里啊？就是核心模块啊！在这个地方：

```
[root@linux ~]# cd /lib/modules/`uname -r`/kernel/drivers/net
[root@linux net]# modinfo 3c59x.ko
filename:       3c59x.ko
author:         Donald Becker <becker@scyld.com>
description:    3Com 3c59x/3c9xx ethernet driver LK1.1.19 10 Nov 2002
license:        GPL
version:        LK1.1.19 DBFD1C112761D573276AC45
parm:           debug:3c59x debug level (0-6)
.....以下省略.....
```

你可以先以 lsmod 查询各个相关的模块后，再到这个地方来检查，或者是直接以上面这个目录下的档案，配合 lsmod 所显示出来的各个模块名称相对应，以取得各个网络卡的模块名称说！这样了解吗？ OK 那你如何知道你的网络卡卡号呢？很简单啊！不管有没有启动你的网络卡，都可以使用：『 ifconfig eth0 』来查询你的网卡卡号。 如果网络卡已经安装好了，那么请前往固定 IP 上网方式去设定 ifcfg-eth0 这个档案 (如果是新增的网卡，这个档案可能不会存在喔!，需要自己建立)。 好了，万一你的网络卡无法被驱动，不过好在官方有提供相关的原始码时， 您就可以自行编译啦！

编译网络卡驱动程序

如果你买到的网络卡实在太新，导致 Linux 预设核心不支持， 或者您实在是很想要一些官方网站提供的驱动程序的新功能， 那实在就得要自行编译核心啦！ 事实上，如果您要新添购硬件时，请先万查阅一下硬件包装上面是否提及支持 Linux 的字样， 因为有些硬件厂商在推出新硬件时，常常会漏掉 Linux 驱动程序的撰写， 如果包装上面有提到支持的话，那么至少您会获得官方网站所提供的驱动程序原始码啊！ ^_^

鸟哥以我们实验室内一部 P-4 内建 Gigabit 以太网络卡的 CentOS Linux 系统来作介绍， 这部主机的内建以太网络使用的是 Marvell 的 88E8001 芯片，鸟哥到他们网站上面找到了一个 2006/04 释出的最新驱动程序来安装，该原始的文件名称为 install-8_40.tar.bz2 ， 看到 .tar.bz2 就应该知道他是属于原始码，虽然鸟哥这部主机已经正确的捉到网络卡了， 不过，我们还是来更新一下让这个模块成为最新的驱动程序吧！ ^_^

另外，记得啊，要编译就得要有 gcc, make, kernel-devel 等套件才行喔！不要说你忘了！ 赶紧回到基础篇里面看看先！ 假设您已经：

- 前往官方网站取得驱动程序的原始码了；
- 已经在您的系统上面安装了 gcc, make, kernel-devel 等套件；

那接下来的编译步骤是这样的：

1. 解压缩与编译：

假设您下载下来的档案放置在 /root 内的话，那么使用 root 的身份进行如下工作吧：

```
[root@linux ~]# cd /usr/src
[root@linux src]# ln -s kernels/2.6.9-34.0.1.EL-smp-i686/ linux
[root@linux src]# cd ~
```

上面这个步骤很重要！因为驱动程序的模块都会需要找到核心的包含档案与相关函式库， 而一般来说， Linux kernel 2.6 以后的版本，则都会预设将核心原始码放在 /usr/src/kernels/(version) 这个目录内，但一般程序却是假设核心原始码在 /usr/src/linux，因此我们必须要先作个手术，将原始码与 /usr/src/linux 作个连结啊！请注意，那个原始码请依照您的核心版本与 Linux distribution 之公布为准。

```
[root@linux ~]# tar -jxvf install-8_40.tar.bz2
[root@linux ~]# cd DriverInstall
# 此时在该目录下有个 README 的档案，记得看一看，这个档案内会说明很多信息，
# 包括如何编译，以及这个模块所支持的芯片组哩！
[root@linux DriverInstall]# ./install.sh
```

这个模块写的比较人性化，因为他有给予一个可直接安装测试的 script， 所以我们可以直接执行这个指令即可，很单纯吧！ ^_^ 。不过记得， 这些动作请在主机前进行。因为这个 script 会主动的重新启动网络卡喔！ 所以如果你是在网络上联机到此主机上面动作的话，嘿嘿！那可是会失败的！ 这个指令会有很多的选项在里面，请依序选择：

- 1) installation
- y(install)

然后这个 script 便会主动的进行编译、模块卸除以及模块安装到 /lib/modules/`uname -r`/kernel/drivers/net 这个目录中， 并且尝试加载这个模块，以及启动网络卡喔！如果一切无误，您就会看到如下画面：

```
eth0: network connection up using port A
    speed:           1000
    autonegotiation: yes
    duplex mode:     full
    flowctrl:        symmetric
    role:            slave
    irq moderation:  disabled
    scatter-gather:  enabled
    tx-checksum:     enabled
    rx-checksum:     enabled
```

```
    rx-polling:      enabled
```

这样就算安装妥当啰！很简单吧！ ^_^

2. 模块之测试与处理

事实上，刚刚我们那个模块其实已经主动的加载到我们的系统当中了， 不过，如果您所取得的原始码并没有附上这么一支人性化的 script 的话，那么您就得要自行进行底下的动作了。

```
[root@linux ~]# ls -l
drwxr-xr-x  2 root root   4096 Jul 20 11:57 sk98lin
[root@linux ~]# ls -l sk98lin
-rw-r--r--  1 root root 2666344 Jul 20 11:57 sk98lin.ko
# 上面那个目录假设是我编译出来的模块，里面还会有一个档案喔！那个档案就是
# sk98lin.ko ， 要注意，新的 2.6 版的核心模块扩展名都变成 .ko 的格式了！

[root@linux ~]# cp -a sk98lin /lib/modules/`uname -r`/kernel/drivers/net
# 注意啊！整个网络卡驱动模块就是放在上面的目录中，不要写错了。

[root@linux ~]# depmod -a
# 将所有的模块进行重新分析的工作！建立关连档案

[root@linux ~]# lsmod | grep 'sk98lin'
# 先确认『sk98lin』这个模块不存在，然后就可以开始测试：

[root@linux ~]# modprobe sk98lin
[root@linux ~]# lsmod | grep 'sk98lin'
# 如果有相关的字样跑出来时，就表示这个模块可以顺利的被加载啦！
```

如果可以顺利加载的话，恭喜您，您所自行编译的驱动程序已经搞定啦！ 不过，当有新版本的核心释出时，您也安装了新版的核心，那么上面的动作您就得要再进行一次， 为什么呢？想一想，刚刚您将编译出来的驱动程序模块放在哪里？ 然后编译的时候参考的核心原始码又是在哪里？ 那您就会晓得『为什么』了！ ^_^

3. 设定开机自动启动网络卡模块

我们前面提到，Linux 的网络卡其实仅是一个代号，并不是类似硬盘的装置档案，所以， 我们必须要指定这个代号与模块的对应才行，在 CentOS (Red Hat 系统) 的对应是使用 /etc/modprobe.conf ，至于旧版的 2.4 核心中，则使用的是 /etc/modules.conf ，注意一下您的核心版本。 鸟哥的 CentOS 4.3 使用的是 /etc/modprobe.conf ，所以我就得这么做：

```
[root@linux ~]# vi /etc/modprobe.conf
```

```
# 修改或增加底下这一行吧！
alias eth0 sk98lin

[root@linux ~]# sync; reboot
```

为了测试一下刚刚的设定是否会生效，通常鸟哥都会尝试一次重新开机， 然后开机完成之后观察一下是否有正确的启动网络卡，并观察一下模块加载的情况， 如果一切都顺利，那就太完美了！

4. 尝试设定 IP

等到一切就绪之后，总得试看看这样的网络卡模块是否可以顺利的设定好 IP 吧？ 所以我们先手动给他一个私有 IP 看看先：

```
[root@linux ~]# ifconfig eth0 192.168.1.100
[root@linux ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:01:BA:77:16:52
          inet addr:192.168.1.100  Bcast:192.168.1.255  Mask:255.255.255.0
.....以下省略.....
```

嘿嘿！真的设定妥当哩！然后利用 ping 这个指令去 ping 一下网域内的其它计算机， 看看能不能有响应，就知道您的网络卡是否 OK 的啦！通常是没有问题的啦！

Linux 网络相关设定档案

要开始玩 Linux 网络之前，请您务必了解 Linux 网络相关设定档案放置的目录与文件名！这真的很重要！因为在 Linux 底下修改 IP 只要动到一个档案即可，修改主机名称也只要一个档案，所以只要将这些文件名称记起来，呵呵！未来在进行网络的测试与修订时，真的会事半功倍～完全不需要什么 linuxconf, webmin 等额外工具的帮助， 真的很简单啦！底下赶紧来说一说与 Linux 网络最相关的几个档案档名与他的用途！

- /etc/sysconfig/network
  这个档案主要的功能在于设定『主机名称(HOSTNAME)与启动 Network 与否』！一般来说，如果您变动过这个档案内的主机名称时，请务必重新开机 (reboot)， 因为这样才会让系统上面所有的程序都使用最新设定的主机名称喔！(在 SuSEL 上面，主机名称是记录在 /etc/HOSTNAME 里面的！)

- /etc/sysconfig/network-scrips/ifcfg-eth0
  这个档案的内容即是设定网络卡参数的档案啦！里面可以设定『network, IP, netmask, broadcast, gateway, 开机时的 IP 取得方式(DHCP, static), 是否在开机的时候启动』等等。至于 ifcfg-eth0 指的是第一块网络卡，而第二块网络卡为 ifcfg-eth1 以此类推。(在 SuSE 上面则是使用 /etc/sysconfig/network/ifcfg-eth0)。

- /etc/modprobe.conf
  如果硬件要能动作，当然要核心有支持才行！同时，我们也可以利用外挂的核心模块 (可以将他

想成是驱动程序) 来驱动硬件，而开机时用来设定加载核心模块的档案就是 modprobe.conf 啦。一般来说，目前的 distributions 通常使用模块的方式来编译网络卡驱动程序，所以， 当您安装完毕后，您的网络卡与模块对应通常已经写到这个档案当中啰！

- /etc/resolv.conf
  我们在网络基础里面稍微提到的 DNS 功能是什么？对啦， 就是进行主机名称与 IP 的对应！而 resolv.conf 就是设定 DNS IP (名称解析服务器) 的档案， 常常有人提到『咦！我已经可以 ping 到外部计算机的公共 IP 了，为何输入网址却无法联机？』 通常发生的错误就是这个档案里面的设定不正确啦！请注意！ 通常这个档案可以输入中华电信的 DNS (168.95.1.1)！

- /etc/hosts
  这个档案可以记录计算机的 IP 对应主机的名称或者主机的别名！特别留意的是， 在局域网络内有些服务需要反查 Client 的身份，此时就会动用到主机名称对应 IP 的数据。 但是因为局域网络内使用私有保留 IP ，当然无法进行 Hostname 对应 IP 的行为， 这个时候该服务就会呆呆的等待 30~60 秒～如果可以避过这个反查，或者是直接告诉主机 Hostname 与 IP 的对应，那么就可以节省这个反查的时间了！所以，如此一来省去反查的时间，就可以改善 LAN 内的联机速度了！这个 hosts 就是用来设定 Hostname 对应 IP 的啦！更多的信息请查阅后续服务器章节 DNS 服务器的介绍。

- /etc/services
  这个档案则是记录架构在 TCP/IP 上面的总总协议，包括 http, ftp, ssh, telnet 等等服务所定义的 port number ，都是这个档案所规划出来的。如果您想要自订一个新的协议与 port 的对应，就得要改这个档案了；

- /etc/protocols
  这个档案则是在定义出 IP 封包协议的相关数据，包括 ICMP/TCP/UDP 这方面的封包协议的定义等。

大概知道上面这几个档案后，未来要修改网络参数时，那就太简单了！ 至于网络方面的启动指令的话，可以记得几个简单的指令即可喔！

- /etc/init.d/network restart
  这个 script 最重要！因为可以一口气重新启动整个网络的参数！ 他会主动的去读取所有的网络设定文件，所以可以很快的恢复系统预设的参数值。

- ifup eth0 (ifdown eth0)
  启动或者是关闭某张网络接口。可以透过这个简单的 script 来处理喔！ 这两个 script 会主动到 /etc/sysconfig/network-scripts/ 目录下， 读取适当的设定档来处理啊！ (例如 ifcfg-eth0)。

大概您只要只到这些基本的指令与档案，哈哈！网络参数的设定就太简单啦！不过，最好您还是要了解 shell script ，比较好！因为可以追踪整个网络的设定条件。 why ？这是因为每个 distributions 的设定数据可能都不太相同，不过却都以 /etc/init.d/network 作为启动的 script ， 因此，您只要了解到该档案的内容，很容易就追踪得出来您的设定档所需要的内容呢！对吧！

连上 Internet 的方法：

在前一章局域网络的简易设定里面，我们知道了连上 Internet 的方法有好多种，不过在现今的台湾，主要的联机方法其实是以 (1)学术网络、(2)ADSL 固接与拨接、(3)Cable modem 等方式为主， 此外，由于目前使用 Linux notebook 的使用者大增，而 Notebook 通常是以无线网络来联机的， 所以鸟哥在这里也尝试使用一款无线网络来进行联机设定。 至于传统的 56 Kbps 拨接则因为速度较慢且使用度越来越低，所以在这里就不多做介绍了。

另外请特别留意，底下的说明全部都是假设您的网络卡已经正常驱动的情况下，所以， 万一您的网络卡还是无法启动时，看是要(1)参考前面的说明重新编译一个模块，还是(2)早点放弃， 赶紧买张便宜的网络卡来安装与设定，那就完全依照您的需求啦！

固定 IP 上网方式

所谓的固定 IP 就是指在您的网络参数当中，你只要输入既定的 IP 参数即可。那么这个既定的 IP 来自哪里呢？ 一般来说，他可能来自于：

- 学术网络：由学校单位直接给予的一组 IP 网络参数；
- 固定制 ADSL：向 ISP 申请的一组固定 IP 的网络参数；
- 企业内部或 IP 分享器内部的局域网络：例如企业内使用私有 IP 作为局域网络的联机之用时，那么我们的 Linux 当然也就需要向企业的网管人员申请一组固定的 IP 网络参数啰！

这样清楚吗？也就是说，我们取得的固定 IP 参数并非一定是 public IP 喔！反正就是一组可接受的固定 IP 就是了！ 所以在架设您的环境之前，请先注意所有网络参数的来源正确性啊！ 好了，那么现在假设我的 Linux 主机需要的参数如下所示：

```
Hostname  centos.dm.tsai

IP:       192.168.1.13
Netmask   255.255.255.0
Network   192.168.1.0
Broadcast 192.168.1.255
Gateway   192.168.1.2

DNS IP    168.95.1.1
```

底下我们就分别针对上面的各项设定来进行档案的重新修改啰！

1. 修改主机名称： /etc/sysconfig/network
   修改主机名称真的是很简单！直接在档案内修订即可！

```
[root@linux ~]# vi /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=centos.dm.tsai
```

够简单吧！这样就将主机名称改好了！记得：如果您曾经更动过这个档案， 最好重新开机，以使所有的服务都可以使用到新主机名称。

2. 设定网络参数： /etc/sysconfig/network-scripts/ifcfg-eth0
请记得，这个 ifcfg-eth0 与档案内的 DEVICE 名称设定需相同，并且，在这个档案内的所有设定，基本上就是 bash 的变量设定规则啦！

```
[root@linux ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0                <== 网络卡代号，必须要 ifcfg-eth0 相对应
BOOTPROTO=static           <== 开机协议，有 dhcp 及 static 这里是 static
BROADCAST=192.168.1.255    <== 就是广播地址啰
HWADDR=00:40:D0:13:C3:46   <== 就是网络卡地址
IPADDR=192.168.1.13        <== 就是 IP 啊
NETMASK=255.255.255.0      <== 就是子屏蔽网络
NETWORK=192.168.1.0        <== 就是网域啊！该网段的第一个 IP
GATEWAY=192.168.1.2        <== 就是预设路由！
ONBOOT=yes                 <== 是否开机启动的意思
MTU=1500                   <== 就是最大传输单元的设定值。
#GATEWAYDEV=eth0           <== 主要路由的装置为那个，通常不用设定
```

上面的资料很好理解吧！请注意每个变量(左边的英文)都应该要大写！ 否则我们的 script 会误判！关于 IP 的四个参数 (IPADDR, NETMASK, NETWORK, BROADCAST) 鸟哥在这里就不再多说，要谈的是几个重要的设定值：

- DEVICE：这个设定值后面接的装置代号需要与文件名 (ifcfg-eth0) 那个装置代号相同才行！否则会显示找不到装置名称！
- BOOTPROTO：启动该网络接口时，使用何种协议？ 如果是手动给予 IP 的环境，请输入 static 或 none ，如果是自动取得 IP 的时候， 请输入 dhcp (不要写错字！)
- GATEWAY：代表的是『整个主机系统的 default gateway』， 所以，设定这个项目时，请特别留意！不要有重复设定的情况发生喔！也就是当您有 ifcfg-eth0, ifcfg-eth1.... 等多个档案，只要在其中一个档案设定 GATEWAY 即可
- GATEWAYDEV：如果您不是使用固定的 IP 作为 Gateway ， 而是使用网络装置作为 Gateway (通常 Router 最常有这样的设定)，那也可以使用 GATEWAYDEV 来设定通讯闸装置呢！不过这个设定项目很少使用就是了！
- HWADDR：这个东西就是网络卡的卡号了！有啥用途呢？ 记得以前我们常常在讲，如果有两块一模一样的网络卡存在时，例如在一部主机上面安装两张螃蟹卡， 由于是相同的芯片，所以在 /etc/modprobe.conf 内无法指定出明确的 eth0 与 eth1 的对应 (因为模块使用相同嘛！)，那么哪一张才是 eth0 ？呵呵！大家有福了！ 利用 HWADDR 指定网络卡的卡号，就能够明白的定义出不同网卡的代号啦！很方便吧！

事实上，如果您想要了解每个变量的项目代表的意义时，建议可以参考 /sbin/ifup 这个 script 的

内容， 这个 script 对于每个项目的应用都记录的挺清楚的！ ^_^

3. 启动与关闭网卡：
启动与关闭的方式有两种，底下分别介绍：

```
[root@linux ~]# ifup eth0
[root@linux ~]# ifdown eth0
# 上面的作法是针对 eth0 来进行启动 (ifup) 与关闭 (ifdown) ；

[root@linux ~]# /etc/init.d/network restart
# 针对这部主机的所有网络接口 (包含 lo) 与通讯闸进行重新启动，
# 所以网络会停顿再开喔！
```

就样就能够启动网络卡了！再来赶紧测试观察看看，我们可以直接下达 ifconfig 以及使用 ping 来检查看看喔！

```
[root@linux ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:40:D0:13:C3:46
          inet addr:192.168.1.13  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::240:d0ff:fe13:c346/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:229458 errors:16 dropped:0 overruns:0 frame:0
          TX packets:117415 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:345006035 (329.0 MiB)  TX bytes:7177543 (6.8 MiB)
          Interrupt:5 Base address:0x3e00
# 记得啊！有出现上头那个 IP 的数据才是正确的启动；
# 另外，也注意一下那个 MTU 的数值呐！ ^_^

[root@linux ~]# ping -c 3 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=0.216 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.227 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.222 ms

--- 192.168.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.216/0.221/0.227/0.017 ms, pipe 2
# 注意啊！有出现 ttl 才是正确的响应！如果出现『 Destination Host Unreachable 』
# 表示没有成功的联机到您的 GATEWAY 那表示出问题啦！赶紧检查有无设定错误。
```

一般来说，到此为止您的 IP 设定已经成功啦！不过，偶而您会发现无法找到 yahoo.com.tw 的 IP 那！ 那表示您必须要进行底下的工作！

4. 设定 DNS 的 IP： /etc/resolv.conf

   这个档案重要啦！他会影响到您是否可以查询到主机名称与 IP 的对应喔！ 通常如下的设定就 OK 了！

```
[root@linux ~]# vi /etc/resolv.conf
nameserver 168.95.1.1
nameserver 139.175.10.20
```

   我们以中华电信与 SeedNet 在南部的 DNS 主机之 IP 作为设定的方式！ 请注意一下，如果您不知道您的最接近的 DNS 主机的 IP ，那么直接输入 nameserver 168.95.1.1 这个中华电信的 DNS 主机即可！不过如果您公司内部有设定防止 DNS 的要求封包的防火墙规则时， 那么您就得要请教贵公司的网管单位告知您的 DNS IP 设定啦！然后赶紧测试看看：

```
[root@linux ~]# nslookup tw.yahoo.com
Server:         168.95.1.1
Address:        168.95.1.1#53

Non-authoritative answer:
tw.yahoo.com    canonical name = tw.yahoo-ap1.akadns.net.
Name:   tw.yahoo-ap1.akadns.net
Address: 202.43.195.52
```

   仔细观察，得要出现有 IP 的字样，才算是设定成功喔！

上面这几个步骤需要一步一步来，如果前面失败，后面就不可能成功的！ 所以请不要尚未启动网络前，就使用 nslookup 去追查 IP ，那是『查不到的』啦！ 注意流程啊！得要网络好了，才能够处理其它在网络上面的 DNS 数据，您说是吧！ ^_^

可自动取得 IP 的环境

可自动取得 IP 的环境是怎么回事啊？不是很简单吗？当您在 IP 分享器后头的主机在设定时， 不是都会选择『自动取得 IP 』吗？那就是可自动取得 IP 的环境啦！那么这个自动取得是怎么回事啊？ 也不难了解啦，其实就是『有一部主机提供 DHCP 服务给整个网域内的计算机』就是了！ 例如 IP 分享器就可能是一部 DHCP 主机。那么 DHCP 是啥？ 他是：Dynamic Host Configuration Protocol 的简写，顾名思义，他可以『动态的调整主机的网络参数』的意思。详细的 DHCP 功能我们会在后面的章节说明的。

好了，那么这个方法适合哪些联机的方式呢？大致有这些：

- Cable Modem：就是使用电视缆线进行网络回路联机的方式啊！ 这个方式属于媒体共享式，在小区内比较常见；
- ADSL 多 IP 的 DHCP 制：就鸟哥所知， SeedNet 有推出一种项目， 可以让 ADSL 用户以 DHCP 的方式来自动取得 IP ，不需要拨接。那使用的也是这种方法！

- IP 分享器或 NAT 有架设 DHCP 服务时：当您的主机位于 IP 分享器的后端，或者是您的 LAN 当中有 NAT 主机且 NAT 主机有架设 DHCP 服务时， 那取得 IP 的方法也是这样喔！

反正可以自动取得 IP 的条件下，大多数都是使用底下的方法啦！ 请先参考前一小节固定 IP 的联机方法设定好步骤 1~4 的所有项目， 其中第 2 个步骤需要改成底下的样子：

```
[root@linux ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

没盖你喔！只要这三个项目即可，其它的都给他批注 (#) 掉！避免互相干扰！然后给他重新启动网络：

```
[root@linux ~]# /etc/init.d/network restart
```

我们局域网络内的 IP 分享器或 DHCP 主机，就会立刻帮您的 Linux 主机做好网络参数的规划， 包括 IP 参数与 GATEWAY 等，就通通设定妥当啦！很方便也很简单吧！

Tips:
有些版本会主动的修改 /etc/resolv.conf 这个档案，所以您必须要随时注意一下这个档案的内容， 另外， CentOS 会主动的建立一个说明文件数据在 /etc/dhclient-eth0.conf ，您也可以自行瞧一瞧去！



ADSL 拨接上网

终于来到台湾最热门的 ADSL 拨接上网的介绍啦！来谈一谈如何在 Linux 上拨接上网吧！ 要拨接上网时，可以使用 rp-pppoe 这套软件来帮忙，所以，您必须要确认你的 Linux distributions 上面已经存在这个玩意儿了！(注： SuSE 使用的是其它的软件，您应该要自行使用 yast 来设定即可。) CentOS 本身就含有 rp-pppoe ，请使用原版光盘，或者是使用 yum 来进行安装吧！

```
[root@linux ~]# rpm -q rp-pppoe
rp-pppoe-3.5-22     <==您瞧瞧！确实有安装喔！
```

当然，很多 distributions 都已经将拨接这个动作归类到图形接口里面去了，所以可能没有提供 rp-pppoe 这个咚咚，没关系，您可以到底下的网站去取得的：

- http://www.roaringpenguin.com/pppoe/
- http://freshmeat.net/projects/rp-pppoe/

然后再自行手动安装即可。如何安装的过程鸟哥在这里就不谈了，请自行前往基础篇的原始码与 Tarball查阅相关资料吧。 另外请注意，虽然整个联机是由主机的以太网络卡连接到 ADSL 调制解调器上，然后再透过电话线路联机到 ISP 的机房去，最后在主机上以 rp-pppoe 拨接达成联机。但是 rp-pppoe 使用的是 Point to Point (ppp) 的点对点协议所产生的网络接口，因此当您顺利的拨接成功之后， 会多产生一个实体网络接口『 ppp0 』喔！ 而由于 ppp0 是架构在以太网络上的，所以 eth0 也不能关掉啊！注意注意！因此，拨接成功后就会有：

- 内部循环测试用的 lo 接口；

- 网络卡 eth0 这个接口；
- 拨接之后产生的经由 ISP 对外连接的 ppp0 接口。

上头这三个接口是完全独立的，互不相干，所以关于 eth0 您可以这样思考：

- 这张网络卡 (假设是 eth0) 有接内部网络(LAN)：
  如果是这样的话，那么您的 IP 设定参数： /etc/sysconfig/network-scripts/ifcfg-eth0 应该要给予一个私有 IP 以使内部的 LAN 也可以透过 eth0 来进行联机啊！所以我会这样设定：

```
[root@linux ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.1.255
IPADDR=192.168.1.13
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
```

  并请记得一件事情，那就是：『千万不要有 GATEWAY 的设定！』， 否则 ppp0 拨接成功后又会产生另一个 default gateway ，就会造成问题了！

- 没有内部网域：
  如果这部 Linux 主机是直接连接到 ADSL 上头，并没有任何内部主机与其联机， 也就是说，您的 eth0 有没有 IP 都没有关系时，那么上面的设定当中的那个『 ONBOOT=yes 』 直接改成『 ONBOOT=no 』就好了！

至于其它的档案请参考前两小节固定 IP 的联机方法上面的步骤 1, 3, 4 设定好！步骤 2 则依照刚刚提到的是否有接 LAN 的情况来设定，然后就可以开始准备拨接了！ 当然啦，拨接之前，请确认您的 ADSL 调制解调器 (小乌龟) 已经与主机联机妥当， 也取得账号与密码，也安装好了 rp-pppoe ，然后就来处理吧！

1. 关闭接在 ADSL 调制解调器上面的那张网卡
   说实在的，鸟哥比较建议将内外网域分的清清楚楚比较好，所以，通常我都是主机上面接两块网络卡， 一张对内一张对外，对外的那张网卡在拨接前先关闭，虽然没有关闭也不打紧， 不过曾有网友回报，如果不关闭时，有时候(机率很低)会无法拨接成功！假设连接到 ADSL 的网卡是 eth0 ，那么关闭很简单吧？

```
[root@linux ~]# ifdown eth0
[root@linux ~]# ifconfig eth0 down
```

   这样就可以啦！

2. 设定账号与密码之联机设定
   假设 rp-pppoe 设定好了，如果您的版本没有 rp-pppoe 的话，请自行安装。 关于安装的方法可以参考本章最后的参考资料。然后就开始如下的设定：

```
[root@linux ~]# adsl-setup
Welcome to the ADSL client setup.  First, I will run some checks on
your system to make sure the PPPoE client is installed properly...

LOGIN NAME
Enter your Login Name (default root): T1234567
# 注意啊！这个账号名称是 ISP 给的，其中如果是 SeedNet ，输入如上，
# 如果是 Hinet 的话，就得要输入 username@hinet.com.tw ，后面的主机名也要写。

INTERFACE
Enter the Ethernet interface connected to the ADSL modem
For Solaris, this is likely to be something like /dev/hme0.
For Linux, it will be ethX, where 'X' is a number.
(default eth0): eth0
# 就是连接到 ADSL 调制解调器的那张网卡代号

Do you want the link to come up on demand, or stay up continuously?
If you want it to come up on demand, enter the idle time in seconds
after which the link should be dropped.  If you want the link to
stay up permanently, enter 'no' (two letters, lower-case.)
NOTE: Demand-activated links do not interact well with dynamic IP
addresses.  You may have some problems with demand-activated links.
Enter the demand value (default no): <==这里按 Enter 确定不要即可

DNS
Enter the DNS information here: 168.95.1.1
Enter the secondary DNS server address here: 139.175.10.20
# 这两个设定会影响 /etc/resolv.conf 的内容喔！

PASSWORD
Please enter your Password: <==这里则是输入您 ISP 给的密码
Please re-enter your Password: <==再一次密码

USERCTRL
Please enter 'yes' (two letters, lower-case.) if you want to allow
normal user to start or stop DSL connection (default yes): no
# 不让使用者可以启动或关闭 ADSL 的设定比较好吧！

The firewall choices are:
0 - NONE: This script will not set any firewall rules.  You are responsible
          for ensuring the security of your machine.  You are STRONGLY
          recommended to use some kind of firewall rules.
1 - STANDALONE: Appropriate for a basic stand-alone web-surfing workstation
```

```
2 - MASQUERADE: Appropriate for a machine acting as an Internet gateway
                for a LAN
Choose a type of firewall (0-2): 0
# 先不要密码！后面我们会使用 Linux 本机的防火墙！

Start this connection at boot time
Do you want to start this connection at boot time?
Please enter no or yes (default no):yes
# 是否要开机的时候就拨接？

Ethernet Interface: eth0
User name:          T1234567
Activate-on-demand: No
Primary DNS:        168.95.1.1
Secondary DNS:      139.175.10.20
Firewalling:        NONE
User Control:       no
Accept these settings and adjust configuration files (y/n)? y
# 如果没有问题就按下 y 开始写入设定档吧！
Adjusting /etc/sysconfig/network-scripts/ifcfg-ppp0
Adjusting /etc/resolv.conf
  (But first backing it up to /etc/resolv.conf.bak)
Adjusting /etc/ppp/chap-secrets and /etc/ppp/pap-secrets
  (But first backing it up to /etc/ppp/chap-secrets.bak)
  (But first backing it up to /etc/ppp/pap-secrets.bak)
# 上面的档案很简单吧！
# ifcfg-ppp0  ：亦即是 ppp0 这个网络接口的设定档案；
# resolv.conf ：这个档案会被备份后，然后以刚刚我们上面输入的数据取代；
# pap-secrets, chap-secrets：我们输入的密码就放在这里！
```

哈哈！这样设定就成功啦！很简单吧！唯一需要注意的是在上面的 username 那个地方，千万注意， 因为 hinet 与 seednet 的设定是不一样的！千万小心呢！否则会无法连上线呦！此外， 由于我们在未来还会有 firewall 的建置，所以这里不太需要使用到防火墙啦！否则也可能无法连上 Internet 呦！另外，注意一下，我们上面使用 adsl-setup 的设定，其实最主要是修改两个档案， 分别是 /etc/ppp/pppoe.conf 这个主要设定档，以及 /etc/ppp/chap-secrets 这个密码储存文件！ 仔细去察看一下 chap-secrets 这个档案的内容，您就可以知道您的密码是否输入错误了！ ^_^

3. 开始拨接上网

   拨接很简单啊！

```
[root@linux ~]# adsl-start
....Connect!
```

这样就是显示连上 Internet 啦！通常比较容易出问题的地方在于硬件的联机情况， 请先确认所有的硬件联机没有问题喔！通常，如果您使用小乌龟 (ATU-R) 时， 请使用跳线连接网络卡与 ATU-R。另外一个容易出错的地方在于输入的账号与密码， 账号与密码都是您的 ISP 给您的，并且注意大小写(可以到 /etc/ppp/chap-secrets 察看一下是否设定错误？)

4. 察看 IP 啦！
   直接使用 ifconfig ppp0 看看能否得到您的 IP 呢？没错！那就是啦！ ^_^

```
[root@linux ~]# ifconfig ppp0
ppp0      Link encap:Point-to-Point Protocol
          inet addr:211.74.249.38  P-t-P:172.16.11.8  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1492  Metric:1
          RX packets:8356088 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8532063 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
```

5. 特殊状况：
   或许您会发现使用 rp-pppoe 拨接之后，您的 /var/log/messages 老是出现这样的讯息：

```
[root@linux ~]# more /var/log/messages
May 10 18:02:22 testing pppoe[8527]: Bogus PPPoE length field (50)
```

这是什么问题啊？在 RP-PPPOE 的官方讨论区上面提过，这个问题是由于 ISP 提供的讯息长度超过 rp-pppoe 预设的检查长度才会发生的一个警告讯息，事实上这个讯息并不重要，不用理会他也没有关系， 不会影响到 Internet 的运作。但是，这样的讯息在 /var/log/messages 里面每一分钟就出现一次， 实在很讨厌！怎么办，可以将他拿掉吗？！当然可以！您可以下载新的 Tarball 来重新编译过！ 详细的步骤是这样的：

步骤一：下载新的 tarball (鸟哥这里以 3.5 版为例)：
到底下的网站下载新的版本：
http://www.roaringpenguin.com/pppoe/#download

步骤二：在 /usr/local/src 底下解开档案，则造成 /usr/local/src/rp-pppoe-3.5/ 目录

步骤三：到 /usr/local/src/rp-pppoe-3.5/src 底下去，修改这四个档案：
* discovery.c
* pppoe.c
* pppoe-server.c
* relay.c
找到这四个档案内容如下代码：

```
  /* Check length */
```

```
if (ntohs(packet.length) + HDR_SIZE > len) {
    syslog(LOG_ERR, "Bogus PPPoE length field (%u)",
           (unsigned int) ntohs(packet.length));
    return;
}
```

将他修改成为如下：

```
/* Check length */
if (ntohs(packet.length) + HDR_SIZE > len) {
    /* syslog(LOG_ERR, "Bogus PPPoE length field (%u)",
           (unsigned int) ntohs(packet.length)); */
    return;
}
```

请注意，上面每个档案都有多个同样的字符串，请依序一个一个都修改掉才行！还没完喔！再到 pppoe.c 找到底下的字眼：

```
if (plen + HDR_SIZE > len) {
    syslog(LOG_ERR, "Bogus length field in session packet %d (%d)",
           (int) plen, (int) len);
    return;
}
```

请将他改成：

```
if (plen + HDR_SIZE > len) {
    /* syslog(LOG_ERR, "Bogus length field in session packet %d (%d)",
           (int) plen, (int) len); */
    return;
}
```

步骤四：然后开始编译与安装吧！

```
[root@linux ~]# cd /usr/local/src/rp-pppoe-3.5/src
[root@linux src]# ./configure
[root@linux src]# make
[root@linux src]# make install
```

当然，您得先移除 rp-pppoe 才行喔！ ^_^

步骤五：利用 adsl-setup ,adsl-start 再重新设定与启动看看吧！

很快的，这样您就已经做好 ADSL 拨接上网的动作了！很快乐吧！ 但是不要忘记了，现在您的主机可是没有任何防备的喔！所以，赶紧往下两节读一读去啊！ ^_^

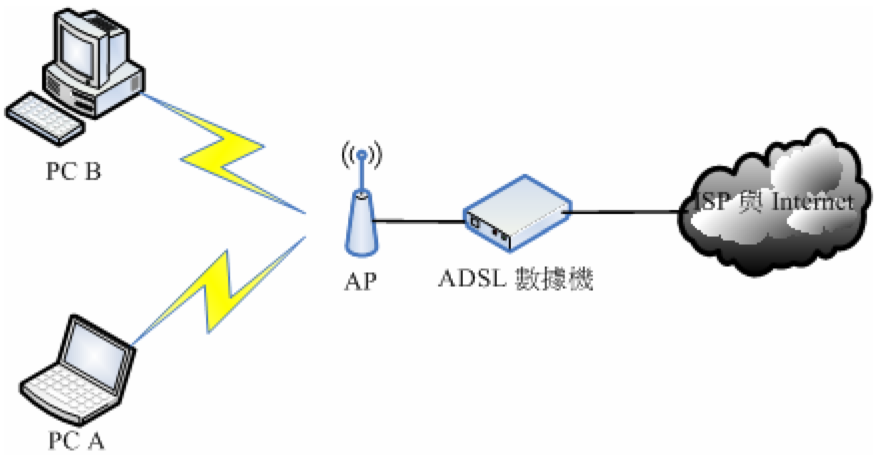
无线网络--以笔记型计算机为例

除了使用实体 RJ-45 线路来连接主机之外，由于现在笔记型计算机渐渐广为使用， 因此在笔记型计算机上面的无线网络 (Wireless) 也越来越重要啰～针对无线网络所提出的标准中， 以 IEEE 802.11b 及 802.11g 较为重要，其中 802.11g 这个标准的传输速度已经可以达到 54Mbps 的水平， 等于是快速以太网络 (fast ethernet, 10/100 Mbps) 的一半了，比起以前的 11Mbps (802.11b) 要快的多！所以也渐渐的成为移动式装置之一的笔记型计算机常用的网络联机方式之一。

无线网络所需要的硬件

我们知道在 RJ-45 的以太网络环境中，以 switch/hub 以及网络卡与网络线最重要， 该架构中主要以 switch/hub 串接所有的网络设备。那么在无线网络中，当然也需要一个接收讯号的装置， 那就是无线基地台 (Wireless Access Point, 简称 AP) 了！ 另一个装置当然就是安装在计算机主机上面的无线网卡啰！

其实无线基地台本身就是个 IP 分享器了，他本身会有两个接口，一个可以与外部的 IP 做沟通， 另外一个则是作为 LAN 内部其它主机的 GATEWAY 啰！那其它主机上面只要安装了无线网卡， 并且顺利的连上 AP 后，自然就可以透过 AP (以 AP 为路由器) 来连上 Internet 啦！ 整个传输的情况可以用下图来示意：



图一、无线网络的联机图标

在上图中，我们假设 PC A 与 PC B 这两部主机都有安装无线网卡，因此他们可以扫瞄到局域网络内的 AP 存在，所以可以透过这个 AP 来连上 Internet 啊。在不考虑内部 LAN 联机的情况下， AP 如何连上 Internet 呢？虽然每部 AP 的控制接口都不相同，不过绝大部分的 AP 都是提供 Web 接口来设定的，因此您可以参考每部 AP 的说明书来进行设定， 在这里鸟哥就不多说了。

鸟哥底下将会以一般家庭常用的小型 AP 与无线网卡来说明一个案例， 鸟哥手边有的笔记型计算机是旧式的 MiTac 内有赛扬 366 CPU 及 192 MB 的内存， USB 版本为 1.1 ，够旧了吧！所以鸟哥额外加了

扩充卡 (PCMCIA) 让 USB 提升到 2.0 版， 选择的 AP 为 PCi 这家公司制造的 BLW-54PM 无线基地台， 而无线网卡则是使用 USB 2.0 接口的 PCi 公司制造的 GW-US54Mini ， 会选择这个装置主要是因为 PCi 这一款无线网卡装置有支持 Linux 系统， 所以很容易被 Linux 捉到，较容易安装使用啦！ ^_^

网络安全方面

如果您留心一下上面的图一，那么就可以发现一件事情，那就是：『如果 AP 不设定任何联机限制，那任何拥有无线网卡的主机都可以透过这个 AP 连接上您的 LAN 』， 要知道，通常我们都会认为 LAN 是信任网域，所以内部是没有防火墙的， 亦即是不设防的状态，呵呵！如果刚好有人拿着笔记型计算机经过您的 AP 可 以接收讯号的范围，那么他就可以轻易的透过您的 AP 连接上你的 LAN ，并且可以透过你的 AP 连上 Internet ，如果他刚好是个喜欢搞破坏的 cracker ， 哈哈！那么当他使用您的 AP 去攻击别人时，最后被发现的跳板是谁？？当然是您的 AP！ 那是谁会吃上官司？够清楚了吧？而且您内部主机的数据也很有可能被窃取啊！

所以啦，『无线网络的安全性一定是具有很大的漏洞的』， 没办法，因为无线网络的传输并不是透过实体的网络线，而是透过无线讯号，实体网络线很好控制， 无线讯号您如何侦测啊？对吧！因此，请您务必在您的 AP 上面进行好联机的限制设定， 一般可以这样做限制的：

- 在 AP 上面使用网卡卡号 (MAC) 来作为是否可以存取 AP 的限制：
  如此一来，就只有你允许的网络卡才能够存取你的 AP ，当然会安全不少。不过这个方法有个问题， 那就是当有其它主机想要透过这个 AP 联机时，你就得要手动的登入 AP 去进行 MAC 的设定， 在经常有变动性装置的环境中 (例如公司行号)，这个方法比较麻烦～
- 设定你的 AP 联机加密机制与金钥：
  另一个比较可行的办法就是设定联机时所需要的验证金钥！这个金钥不但可以在网络联机的数据当中加密，使得即使您的数据被窃听， 对方也是仅能得到一堆乱码，同时由于 client 端也需要知道金钥并且在联机阶段输入金钥， 因此也可以被用来限制可联机的用户啊！

当然，上面两种方法您可以同时设定，亦即不但需要联机的金钥，而且在 AP 处也设定能够存取的 MAC 网卡， 嘿嘿！这样一来，就更安全的多了！更多的 AP 安全方面的概念，可以参考底下这一篇文章的介绍：

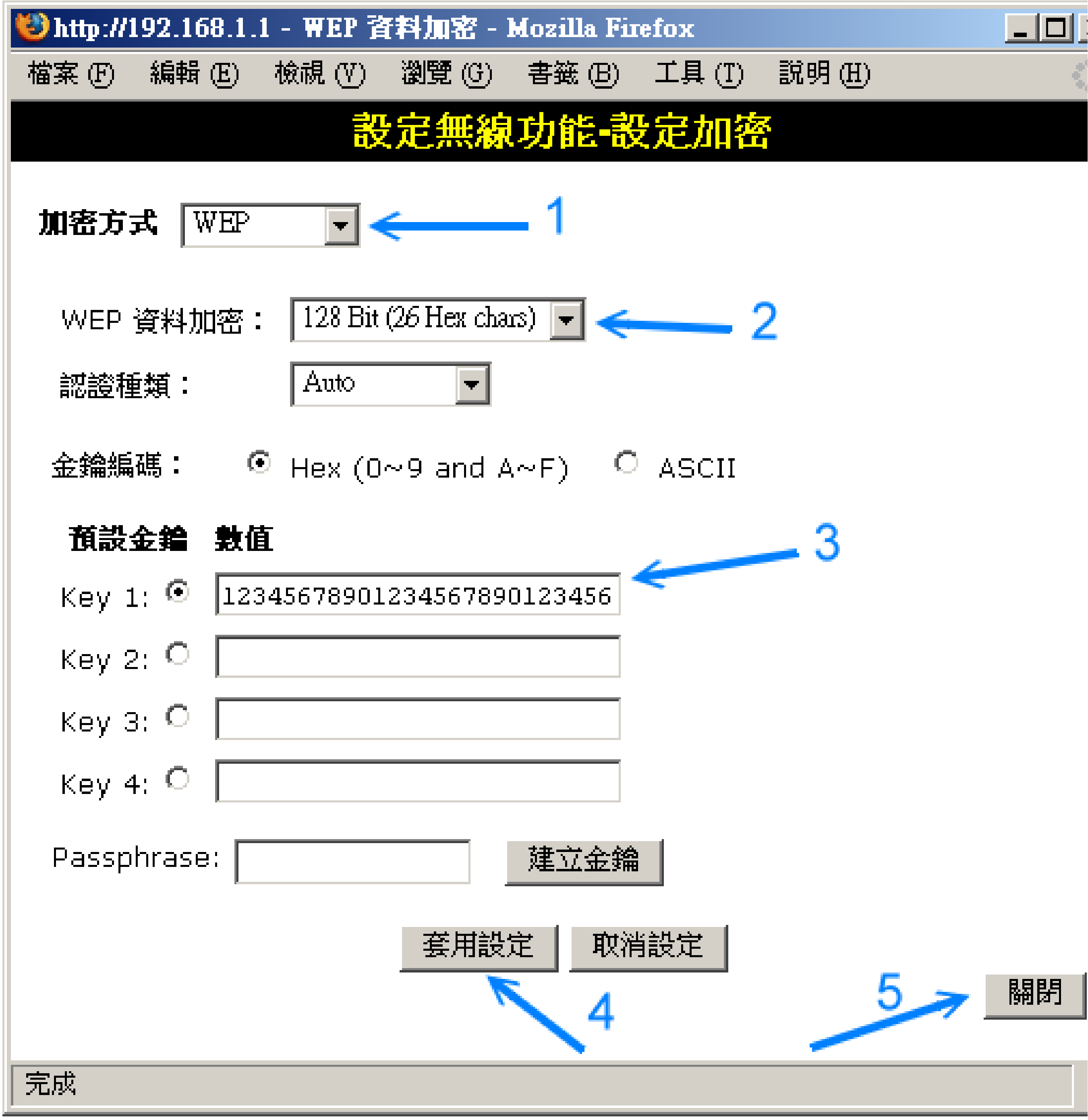
- 无线网络安全白皮书(三)：
  http://www.cert.org.tw/document/column/show.php?key=61

关于 ESSID/SSID ：
想一想，如果您有两部 AP 在同一个局域网络内，那么请问一下， 当你的无线网卡在上网时，他会透过哪一个 AP 联机出去呢？很困扰，对吧！ 其实每部 AP 都会有一个联机的名字，那就是 ESSID ，这个 ESSID 可以提供给 client 端， 当 client 端需要进行无线联机时，他必须要说明他要利用哪一部 AP ，那个 ESSID 就是那时需要输入的数据了！在鸟哥的案例当中，我将我的 AP 设定为 BLW-VBIRD 这个名字，并且给予一个金钥密码，设定的方法如同下图所示：



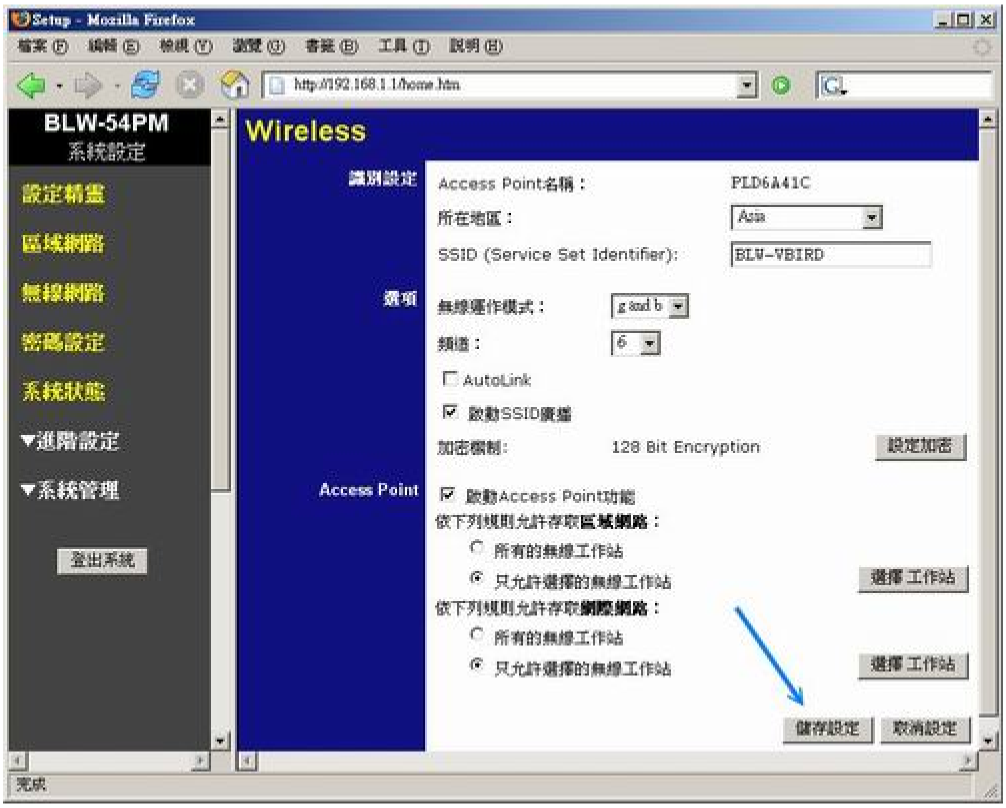
图二、无线网络 AP 的金钥设定项目

如上图，在登入了 AP 的设定项目后，依序 (1)先选择无线网络，然后在右边的窗口当中 (2)取消 AutoLink 的设定，然后 (3)输入您的 SSID 识别码 (就是 ESSID)，最后再进入 (4)密码设定的项目啰！在按下『设定加密』之后，他就会出现如下的画面：



图三、无线网络 AP 的金钥设定项目

我们先选择 (1)加密的方式，在这里可以选择简单的 WEP 加密方式即可，然后 (2)输入加密的金钥长度，选择最长的吧！这个长度需要输入 26 个字符呐！之后当然就得 (3)输入 26 个密码 (金钥) 啰～在这里鸟哥随便先填一个，您可别真的跟鸟哥填一样的密码啊！ 填完之后就 (4)按下套用设定，最后才 (5)关闭，关闭回到前一个画面后， 给他按下『储存设定』，如下图所示：



图四、无线网络 AP 的金钥设定项目

这个时候在画面当中的『加密机制』才会正式的启动啊！这个时候我们就会有底下两个数据：

- ESSID 为 BLW-VBIRD
- KEY 为 12345678901234567890123456

这两个数据很重要喔！我们底下会仔细的来说明啊！

开始联机

OK！底下我们就来谈一谈，那么您的笔记型计算机如何透过无线网络实际的上线呢？ 首先当然就是需要安装驱动程序啦！再来则是需要让网卡代号与模块对上关系， 然后设定了网络卡设定文件后，才能够连上 Internet 啦！底下我们就一个一个步骤来说明吧！

---

1. 取得驱动程序并实际安装：

   鸟哥在这个案例当中使用的 USB 无线网卡的驱动程序，并没有在 CentOS 的预设核心内支持，所以我必须要自行安装他的驱动程序才行！先来察看一下我们的主机是否有捉到这个 USB 的硬件装置呢？

```
[root@linux ~]# dmesg | grep usb
usb 4-1: new high speed USB device using address 2
```

```
# 这里是说，有个高速 USB 装置使用第二个 USB 地址的意思。
# 既然知道这个地址，我们来查一查这个装置的内容吧！

[root@linux ~]# cat /proc/bus/usb/devices
T:  Bus=04 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#=  2 Spd=480 MxCh= 0
D:  Ver= 2.00 Cls=ff(vend.) Sub=ff Prot=ff MxPS=64 #Cfgs=  1
P:  Vendor=14ea ProdID=ab13 Rev=43.30
S:  Manufacturer=PCI
S:  Product=USB2.0 WLAN
.....后面省略.....
# 比较重要的就是上面的那个装置码以及厂商与产品代码了！
# 先给他抄写下来，因为等一下我们进行模块编译时会用到！
```

呵呵！真的有捉到！那么就得要实际的来进行驱动程序的安装啰！ 这个驱动程序的下载可以前往他的官方网站： http://www.planex.com.tw/download/wireless/gw-us54mini.htm 来下载， 鸟哥下载的是 zd1211 的驱动程序，版本是 V1.2.0.0 ，这个版本在编译时， 似乎还是没有做得很好，所以我们依旧得要进行一些小手术来修订才行喔！ 假设您将下载的 tarball 解压缩到 /root/zd1211 档案当中了，那么如何编译呢？

```
[root@linux ~]# cd zd1211
# 这个目录下有个 README 的档案，请记得好好查阅一下喔！

[root@linux zd1211]# cd zdsta
[root@linux zdsta]# vi Makefile
# 先修改底下这一行，应该是在第 15 行的地方，这里需要与你的核心对应喔！
# 我这里是使用 CentOS 4.3 并且升级过核心作为范例的！
KERNEL_SOURCE=/lib/modules/2.6.9-34.0.2.EL/build

[root@linux zdsta]# vi src/zdusb.h
# 很奇怪，我的 CentOS 4.3 不支持太长的型号名称，所以需要做个修改，
# 大约在第 16 行的地方，找到下列的字：
#define PRODUCT_GW-US54MINI    0xAB13
# 将他改成下列的模样：
#define PRODUCT_US54MINI    0xAB13
# 顺便仔细看一下第 15, 16 行，有没有发现版本与型号最右侧，
# 与我们刚刚使用 cat /proc/bus/usb/devices 的内容相同啊！ ^_^

[root@linux zdsta]# vi src/zdusb.c
# 找到 63 行左右的地方，如下所示：
    { USB_DEVICE(VENDOR_PLANEX, PRODUCT_GW-US54MINI )  },
# 将他改成与上述档案相同的型号数据啊！
    { USB_DEVICE(VENDOR_PLANEX, PRODUCT_US54MINI )  },
```

这样就做好了编译前准备，先注意一下，在编译的过程当中会出现很多的 warning ， 不过并不会影响最终的结果，所以就不要理他没关系啊！

```
[root@linux zdsta]# make
# 编译完成后会产生 zd1211.ko 的模块档案喔！
[root@linux zdsta]# make install
# 可以将模块安装到核心模块放置的地方去了！
```

---

2. 测试模块，并且对应网络卡与模块：(modprobe 与 iwconfig)

编译好模块后当然就是要测试看看的啦！这个时候请这样做：

```
[root@linux ~]# modprobe zd1211
[root@linux ~]# lsmod | grep zd1211
zd1211                 226768  0
# 是的！有啊！确实有载入啊！

[root@linux ~]# iwconfig
eth1       802.11b/g NIC  ESSID:""
          Mode:Managed  Frequency=2.462GHz  Access Point: 00:00:00:00:00:00
          Bit Rate:1Mb/s
          Retry:off   RTS thr=2432 B   Fragment thr:off
          Encryption key:off
          Power Management:off
```

这个 iwconfig 是用在作为无线网络设定之用的一个指令，与 ifconfig 类似！ 不过，当我们使用 iwconfig 时，如果有发现上述的字样，那就代表该网络接口使用的是无线网卡的意思啊！ 所以，俺的无线网卡代号是 eth1 喔！之后，我将这个模块与网卡的代号写入 /etc/modprobe.conf 当中吧！

```
[root@linux ~]# vi /etc/modprobe.conf
alias eth1 zd1211
将上面这一行新增到您的 Linux 当中啊！
```

---

3. 利用 iwlist 侦测 AP：

好了，接下来要干嘛？当然是看看我们的无线网卡是否能够找到 AP 啊！ 所以，首先我们要启动无线网卡，就利用 ifconfig 即可：

```
[root@linux ~]# ifconfig eth1 up
```

启动网卡后才能以这个网卡来搜寻整个区域内的无线基地台啊！接下来， 直接使用 iwlist 来使用这个无线网卡搜寻看看吧！

```
[root@linux ~]# iwlist eth1 scanning
eth1      Scan completed :
          Cell 01 - Address: 00:90:CC:D6:A4:1C
                    ESSID:"BLW-VBIRD"
                    Mode:Master
                    Frequency:2.437GHz (Channel 6)
                    Quality:8/92  Signal level=54/154  Noise level=0/154
                    Encryption key:on
                    Bit Rate:1Mb/s
                    Bit Rate:2Mb/s
                    Bit Rate:5.5Mb/s
                    Bit Rate:11Mb/s
                    Bit Rate:6Mb/s
                    Bit Rate:9Mb/s
                    Bit Rate:12Mb/s
                    Bit Rate:18Mb/s
                    Bit Rate:24Mb/s
                    Bit Rate:36Mb/s
                    Bit Rate:48Mb/s
                    Bit Rate:54Mb/s
```

注意到上头的显示喔！有显示 ESSID 没错吧！这个东西等一下可是需要设定的喔！ 其实接下来我们可以直接使用手动的方式来启动我们的无线网卡的联机喔！

```
[root@linux ~]# iwconfig eth1 essid "BLW-VBIRD"  \
> key "12345678901234567890123456"

[root@linux ~]# iwconfig eth1
eth1      802.11b/g NIC  ESSID:"BLW-VBIRD"
          Mode:Managed  Frequency=2.437GHz  Access Point: 00:90:CC:D6:A4:1C
          Bit Rate:11Mb/s
          Retry:off   RTS thr=2432 B   Fragment thr:off
          Encryption key:****-****-****-****-****-****-**   Security mode:open
          Power Management:off
```

如果顺利出现上面的数据，那就表示您的无线网卡已经与 AP 接上线了～ 不过有个地方比较奇怪，怎么最高传输速率仅有 11Mb/s 啊？太低了～ 没关系，我们可以在底下进行处理的。 再来则是设定网络卡的设定文件啰！^_^

---

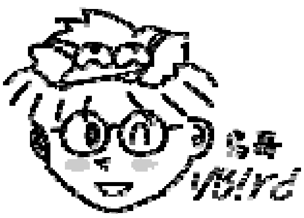
4. 设定网络卡设定文件 (ifcfg-ethn)

因为我们的网络卡使用的代号是 eth1 ，所以也是需要在 /etc/sysconfig/network-scripts 设定好相对应的档案才行啊！而由于我们的这块卡其实是无线网卡， 所以很多设定值都与原本的以太网络卡不同，详细的各项变量设定您可以自行参考一下底下的档案：
/etc/sysconfig/network-scripts/ifup-wireless
至于我的网络卡设定是这样的：

```
[root@linux ~]# cd /etc/sysconfig/network-scripts
[root@linux network-scripts]# vi ifcfg-eth1
DEVICE=eth1
BOOTPROTO=dhcp
ONBOOT=no
TYPE=wireless
ESSID=BLW-VBIRD
MODE=Managed
RATE=54M      <== 可以严格指定传输的速率，要与上面 iwconfig 相同，单位 b/s
KEY=12345678901234567890123456
```

要注意的是那个 ONBOOT=no 的设定，如果您想要每次开机时无线，网卡都会自动启动， 那就将他设定为 yes 吧！否则就设定为 no 啰！要启动再以 ifup eth1 来启动即可！ 呼呼！到此为止，您的无线网卡已经可以顺利的给他启动了喔！很快乐吧！ ^_^

Tips:
其实上面那个设定档的内容都是在规划出 iwconfig 的参数而已，所以您除了可以查阅 ifup-wireless 的内容外， 可以 man iwconfig ， 会知道的更详细喔！而最重要的参数当然就是 ESSID 及 KEY 啰！ ^_^



---

5. 启动与观察无线网卡

要启动就用 ifup eth1 来启动，很简单啦！要观察就用 iwconfig 及 ifconfig 分别观察， 底下您自己瞧瞧就好啊！ ^_^

```
[root@linux ~]# ifup eth1
```

```
# 如果上面的测试都没有问题的话，那么建议您继续底下的动作

[root@linux ~]# iwconfig eth1
eth1      802.11b/g NIC  ESSID:"BLW-VBIRD"
          Mode:Managed  Frequency=2.437GHz  Access Point: 00:90:CC:D6:A4:1C
          Bit Rate:54Mb/s
          Retry:off   RTS thr=2432 B   Fragment thr:off
          Encryption key:****-****-****-****-****-****-**   Security mode:open
          Power Management:off

[root@linux ~]# vi /etc/rc.d/rc.local
# 加入底下这段：
/sbin/ifconfig eth1 up
```

整个流程就是这么简单喔！不过，如果您的笔记型计算机已经有支持 802.11g/b 的无线网卡时， 比如说使用 Intel 规格的迅驰 (Centrino) 笔记型计算机，那除了上面在加载魔组的地方不一样， 因为 CentOS 4.3 本身就有支持 Centrino 的模块，那就是 ipw2200/ipw2100 ， 所以您可以直接跳到第二步，甚至可能在安装的时候系统就主动的帮您安装好这个无线网卡了呢！ 那您就可以直接前往第三步骤开始设计您的 AP 与无线网卡的联机啰！ ^_^。在本章结尾的参考资料处， 鸟哥还是列出许多与无线网卡有关的连结，您可以自行前往查阅与您的无线网卡有关的信息喔！ ^_^

常见问题说明

其实这个小节也很重要的！因为可以让您在念完理论后，了解一下如何利用那些概念来查询您的网络设定问题喔！ 底下我们就针对几个常见的问题来说说看吧！

内部网域使用某些联机服务(如 FTP, POP3)所遇到的联机延迟问题

您或许曾经听过这样的问题：『我在我的虚拟网域内有几部计算机， 这几部计算机明明都是在同一个网域之内，而且系统通通没有问题，为什么我使用 pop3 或者是 ftp 连上我的 Linux 主机会停顿好久才连上？』

由于网络在联机时，两部主机之间会互相询问对方的主机名称，以确认对方的身份。 在目前的因特网上面，我们大多使用 Domain Name System (DNS) 系统做为主机名称与 IP 对应的查询，那就是我们在上面提到的 /etc/resolv.conf 档案内设定的 IP 由来， 如果没有指定正确的 DNS IP 的话，那么我们就无法查询到主机名称与 IP 的对应了。

公开的因特网可以这样设定，但是如果是我们内部网域的私有 IP 主机呢？ 因为是私有 IP 的主机，所以当然无法使用 /etc/resolv.conf 的设定来查询到这部主机的名称啊！ 那怎么办？要知道，如果两部主机之间无法查询到正确的主机名称与 IP 的对应， 那么将『可能』发生持续查询主机名称对应的动作，这个动作一般需要持续 30-60 秒，因此，您的该次联机将会持续检查主机名 30 秒钟，也就会造成奇怪的 delay 的情况。

这个问题最常发生在内部的 LAN ，例如使用 192.168.10.1 的主机联机到 192.168.10.2 的主机。 这个问

题虽然可以透过修改软件的设定来略过主机名称的检查，但是绝大多数的软件都是预设启用这个机制的，因此，内部主机『老是联机时期很慢，联机成功后速度就会恢复正常』 时，通常就是这个问题啦！尤其是在 FTP 及 POP3 等网络联机软件上最常见。

那么如何避过这个情况？最简单的方法就是『给予内部的主机每部主机一个名称与 IP 的对应』即可。举例来说，我们知道每部主机都有一个主机名称为 localhost ，对应到 127.0.0.1 ，为什么呢？因为这个 127.0.0.1 与 localhost 的对应就被写到 /etc/hosts 内嘛！ 当我们需要主机名称与 IP 的对应时，系统就会先到 /etc/hosts 找寻对应的设定值， 如果找不到，才会使用 /etc/resolv.conf 的设定去因特网找。这样说，您明白了吧？ 也就是说，只要修改了 /etc/hosts，加入每部主机与 IP 的对应， 就能够避过主机名称的检查啰！

了解了吗？所以说，您就要将您的 私有 IP 的计算机与计算机名称写入您的 /etc/hosts 当中了！好了！那么这个咚咚的内容如何呢？ 我们来看一看原本的设定内容吧！

```
[root@linux ~]# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1             localhost.localdomain    localhost
# 主机的 IP             主机的名称               主机的别名
```

在上面的情况中很容易就发现了设定的方法了吧！很简单吧！没错！那就是 IP 对应主机名称啦！ 那么现在知道为什么我们给他 ping localhost 的时候，地址会写出 127.0.0.1 了吧！ 那就是写在这个档案中的啦！而且 localhost 那一行不能拿掉呦！ 否则系统的某些服务可能就会无法被启动！好了！那么将我局域网络内的所有的计算机 IP 都给他写进去！并且，每一部给他取一个您喜欢的名字， 即使与 client 的计算机名称设定不同也没关系啦！ 以鸟哥为例，如果我还额外加设了 DHCP 的时候，那么我就干脆将所有的 C Class 的所有网段全部给他写入 /etc/hosts 当中，有点像底下这样：

```
[root@linux ~]# vi /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1             localhost.localdomain    localhost
192.168.1.1     linux001
192.168.1.2     linux002
192.168.1.3     linux003
........
........
192.168.1.255   linux255
```

如此一来，不论我哪一部计算机连上来，不论是在同一个网段的哪一个 IP ， 我都可以很快速的追查到！嘿嘿！那么区内网络互连的时候，就不会多等个好几时秒钟啰！

网址列无法解析问题

很多朋友常问的一个问题『咦！我可以拨接上网了，也可以 ping 到奇摩雅虎的 IP ，但为何就是无法直接以网址连上 Internet 呢！』嘿！被气死！ 前面不是一直强调那个 DNS 解析的问题吗？对啦！就是名称解析不对啦！赶快改一下 /etc/resolv.conf 这个档案吧！改成上层 ISP 给您的 DNS 主机的 IP 就可以啦！例

如 Hinet 的 168.95.1.1 及 Seednet 的 139.175.10.20 啰！例如底下的范例(这个范例就可以照抄了！ ^_^)：

```
[root@linux ~]# vi /etc/resolv.conf
nameserver 168.95.1.1
nameserver 139.175.10.20
```

朋友们常常会在这个地方写错，因为很多书上都说这里要设定成为 NAT 主机的 IP ， 那根本就是不对的！您应该要将所有管理的计算机内，关于 DNS 的设定都直接使用上面的设定值即可！ 除非您的上层环境有使用防火墙，那才另外考虑！

预设路由的问题

记得我们在前两章提到的网络基础当中，不是讲了很多预设路由 (default gateway) 相关的说明吗？ 预设路由通常仅有一个，用来做为同一网域的其它主机传递非本网域的封包网关。 但我们也知道在每个网络设定档案 (/etc/sysconfig/network-scripts/ifcfg-ethx) 内部都可以指定『 GATEWAY 』这个参数，若这个参数重复设定的话，那可就麻烦啦！

举例来说，您的 ifcfg-eth0 用来做为内部网域的沟通，所以您在该档案内设定 GATEWAY 为您自己的 IP ，但是该主机为使用 ADSL 拨接，所以当拨接成功后会产生一个 ppp0 的接口，这个 ppp0 接口也有自己的 default gateway ，好了，那么当你要将封包传送到 Yahoo 这个非为本网域的主机时， 这个封包是要传到 eth0 还是 ppp0 呢？因为两个都有 default gateway 啊！

没错！很多朋友就是这里搞不懂啦！常常会错乱～所以，请注意， 您的 default gateway 应该只能有一个，如果是拨接，请不要在 ifcfg-eth0 当中指定 GATEWAY或 GATEWAYDEV 等变量，重要重要！

更多的网络除错请参考后续章节Linux 网络侦错 的说明。

重点回顾

- Linux 网络卡的预设代号为 eth0, eth1 等等；
- Linux 内的网络卡代号为一个代号，并非为装置档案。欲对应装置代号时， 可在 /etc/modprobe.conf 内制作好网卡代号与驱动模块的对应。
- 若要取得网络卡的完整功能，有时需要自行由网卡开发商的官方网站下载适合的原始码来安装。 编译与安装模块前必须先确定 gcc, make, kernel-devel (核心原始码) 已经安装完毕。
- 内部网域的私有 IP 之主机的『 IP 与主机名称的对应』，最好还是写入 /etc/hosts ， 可以克服很多软件的 IP 反查所花费的等待时间。
- 在 Red Hat base 的 Linux distributions 当中，网络设定档案大多放置于 /etc/sysconfig/network-scripts/ 目录下，尤其是该目录下的 ifcfg-eth0 可设定网络参数；
- 在 ifcfg-eth0 当中，可以指定 MTU 以设定网络卡的最大传输单元，也可以利用 HWADDR 指定出所需要设定的网卡；
- 在 GATEWAY 这个参数的设定上面，务必检查妥当，仅设定一个 GATEWAY 即可。
- 可以使用 /etc/init.d/network restart 来重新启动整个系统的网络接口。

- 若使用 DHCP 协议时，则请将 GATEWAY 取消设定，避免重复出现多个 default gateway ，反而造成无法联机的状况。
- 拨接后可以产生一个新的实体接口，名称为 ppp0
- 主机名称与 IP 的对应，通常使用 DNS 系统，该系统以 /etc/resolv.conf 做为服务器 IP 设定的档案。
- 无线网卡与无线基地台之间的联机由于是透过无线接口，所以需要特别注意网络安全；
- 常见的无线基地台(AP)的联机防护，主要利用控制登入者的 MAC 或者是加上联机加密机制的金钥等方法；
- 设定网络卡可以使用 ifconfig 这个指令，而设定无线网卡则需要 iwconfig ，至于扫瞄基地台，可以使用 iwlist 这个指令。

课后练习

- 我要如何确定我在 Linux 系统上面的网络卡已经被 Linux 捉到并且驱动了？

  网络卡能不能被捉到可以使用『 dmesg|grep eth 』来判断，有没有驱动则可以使用 lsmod 看看模块有没有加载核心！最后，以 ifconfig eth0 192.168.0.10 测试看看！

- 假设我的网络参数为：IP 192.168.100.100, Netmask 255.255.255.0, 请问我要如何在 Linux 上面设定好这些网络参数 (未提及的网络参数请自行定义！)？请使用手动与档案设定方法分别说明。

  手动设定为：『 ifconfig eth0 192.168.100.100 netmask 255.255.255.0 up 』
  档案设定为：vi /etc/sysconfig/network-scripts/ifcfg-eth0 ，内容为：

  DEVICE=eth0
  ONBOOT=yes
  BOOTPROTO=static
  IPADDR=192.168.100.100
  NETMASK=255.255.255.0
  NETWORK=192.168.100.0
  BROADCAST=192.168.100.255 要启动则使用 ifup eth0 即可！

- 我要将我的 Linux 主机名称改名字，步骤应该如何(更改那个档案？如何启用？)?

  Linux 主机名称在 /etc/sysconfig/network 这个档案里面的『HOSTNAME=主机名称』来设定，先以 vi 来修改，改完后可以使用 /etc/init.d/network restart 不过建议直接 reboot 启动主机名称！

- /etc/resolv.conf 与 /etc/hosts 的功能为何？

  以主机名称寻找 IP 的方法， /etc/resolv.conf 内填写 DNS 主机名称，至于 /etc/hosts 则直接填写主机名称对应的 IP 即可！ 其中 /etc/hosts 对于内部私有 IP 的主机名称查询非常有帮助！

- 我使用 ADSL 拨接连上 Internet ，请问拨接成功之后，我的 Linux 上面会有几个网络接口 (假设我只有一个网络卡)？

  因为拨接是使用 PPP (点对点)协议，所以拨接成功后会多出一个 ppp0 的接口，此外，系统原本即有 eth0 及 lo 这两个界面，所以共有三个界面。

- 在 Linux 上面进行 ADSL 拨接应该使用什么软件？

  其实软件非常多，尤其是图形接口的拨接软件，多的很！ 不过，依旧请爱用 rp-pppoe ，官方网站： http://www.roaringpenguin.com/pppoe/

- 一般来说，如果我拨接成功，也取得了 ppp0 这个接口，但是却无法对外联机成功， 您认为应该是哪里出了问题？该如何解决？

  因为拨接成功了，表示物理对外联机没有问题，那么可能的问题应该是发生在 Gateway 上面了！确认的方法请使用 route -n 查阅路由信息，然后修订 /etc/sysconfig/network-scripts/ifcfg-eth0 吧！

- 如果您的局域网络环境内有可以控管的无线 AP 时，请自行查出如何以 MAC 的方式管理可登入的用户， 并将您的无线 AP 做好联机加密的金钥设定。

  请自行测试！谢谢！

- 如果一部主机上面插了两张相同芯片的网络卡，代表两者使用的模块为同一个，此时可能会造成网卡代号的误判； 请问您如何克服这个问题？让网卡代号不会变动？

  以现在的方法来讲，其实我们可以透过指定 Hardware Address(硬件地址，通称为 MAC) 来指定网卡代号与 MAC 的对应。 这个设定值可以在 ifcfg-ethx 里面以 HWADDR 这个设定项目来指定的。

- 如何在 Linux 上面的文字接口搜寻您所在区域的无线 AP ？

  透过直接使用『 iwlist ethx scanning 』这个指令来指定某个无线网卡的搜寻！ 然后再以 iwconfig 来进行网卡的设定即可！

- 请依序说明：如果您想要新增一块新的网络卡在您的主机上，并给予一个固定的私有 IP ，应如何进行？
  - 先关掉主机的 power ，然后拆掉机壳，装上网络卡；
  - 开机完成后，以 dmesg | grep eth 查询是否可捉到该网络卡，若无法捉到，请编译模块，若可捉到，找出网卡代号， 并且将该模块与网卡代号写入 /etc/modprobe.conf 当中，以利未来开机时可自动达成对应；
  - 利用『 ifconfig "网卡代号" 』来查询 MAC 为何？
  - 开始在 /etc/sysconfig/network-scripts 内建立 ifcfg-"网卡代号" 档案，同时给予 HWADDR 的对应；
  - 启动 /etc/init.d/network restart 测试是否能成功！
- 如果您想要登入某个区域的无线 AP ，您应该向该处所至少申请哪些数据？

无线网络的技术相当多且复杂，所以需要取得的参数都不尽相同。不过，至少您还是得要取得 ESSID 以及 KEY 密码，这样才能够联机登入该 AP 当中。

参考数据：

- rp-pppoe 官方网站：http://www.roaringpenguin.com/pppoe/
- rp-pppoe 的安装方法：
  http://linux.vbird.org/linux_server/0130internet_connect/0130internet_connect.php#connect_adsl
- 无线网络安全白皮书(三)：http://www.cert.org.tw/document/column/show.php?key=61
- GW-US54Mini 驱动程序下载：http://www.planex.com.tw/download/wireless/gw-us54mini.htm
- Intel Centrino 的无线网卡相关模块信息：
  http://ipw2100.sourceforge.net/
  http://ipw2200.sourceforge.net/
- HP 的许多无线网络的计划连结：http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/

Linux 常用网络指令介绍

最近更新日期：2006/08/02

Linux 的网络功能相当的强悍，一时之间我们也无法完全的介绍所有的网络指令， 这个章节主要的目的在介绍一些常见的网络指令而已。至于每个指令的详细用途将在后续服务器架设时， 依照指令的相关性来进行说明。当然，在这个章节的主要目的是在于将所有的指令汇整在一起，比较容易了解啦！ 还有，这一章鸟哥新增了一些封包撷取的指令，若不熟悉没关系，先放着，全部读完后再回来这一章仔细练习啊！

网络参数设定指令：

任何时刻如果你想要做好你的网络参数设定，包括 IP 参数、路由参数与无线网络等等， 就得要了解底下这些相关的指令才行！其中以 route 及 ip 这两支指令算是较重要的喔！ ^_^ 当然，比较早期的用法，我们都是使用 ifconfig 的啦！

- ifconfig：查询、设定网络卡与 IP 网域等相关参数；

- ifup, ifdown：这两个档案是 script ，透过更简单的方式来启动网络接口；
- route：查询、设定路由表 (route table)
- ip：复合式的指令，可以直接修改上述提到的功能；

---

ifconfig, ifup, ifdown

这三个指令的用途都是在启动网络接口，不过， ifup 与 ifdown 仅能就 /etc/sysconfig/network-scripts 内的 ifcfg-ethx (x 为数字) 进行启动或关闭的动作，并不能直接修改网络参数，除非手动调整 ifcfg-ethx 档案才行。至于 ifconfig 则可以直接手动给予某个接口 IP 或调整其网络参数！ 底下我们就分别来谈一谈先！

- ifconfig

ifconfig 主要是可以手动的启动、观察与修改网络接口的相关参数，可以修改的参数很多啊， 包括 IP 参数以及 MTU 等等都可以修改，他的语法如下：

```
[root@linux ~]# ifconfig {interface} {up|down}  <== 观察与启动接口
[root@linux ~]# ifconfig interface {options}    <== 设定与修改接口
参数：
interface：网络卡接口代号，包括 eth0, eth1, ppp0 等等
options  ：可以接的参数，包括如下：
    up, down ：启动 (up) 或关闭 (down) 该网络接口(不涉及任何参数)
    mtu      ：可以设定不同的 MTU 数值，例如 mtu 1500 (单位为 byte)
    netmask  ：就是子屏蔽网络；
    broadcast：就是广播地址啊！
范例：

范例一：观察所有的网络接口(直接输入 ifconfig)
[root@linux ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0F:EA:A3:06:A2
          inet addr:192.168.10.100  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20f:eaff:fe73:682/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3439 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2735 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:646935 (631.7 KiB)  TX bytes:562313 (549.1 KiB)
          Interrupt:209 Memory:fb000000-0
```

一般来说，直接输入 ifconfig 就会列出目前已经被启动的卡， 不论这个卡是否有给予 IP ，都会被显示出来。而如果是输入 ifconfig eth0 ， 则会秀出这张接口的相关数据，而不管该接口是否有启动。所以如果您想要知道某张网络卡的 Hardware Address，直接输入『 ifconfig "网络接口代号" 』即可喔！ ^_^！至于上表出现的各项数据是这样的(数据排列由上而下、由左而右)：

- eth0：就是网络卡的代号，也有 lo 这个 loopback ；

- HWaddr：就是网络卡的硬件地址，俗称的 MAC 是也；
- inet addr：IPv4 的 IP 地址，后续的 Bcase, Mask 分别代表的是 Broadcast 与 netmask 喔！
- inet6 addr：是 IPv6 的版本的 IP ，我们没有使用，所以略过；
- MTU：就是 MTU 啊！
- RX：那一行代表的是网络由启动到目前为止的封包接收情况， packets 代表封包数、errors 代表封包发生错误的数量、 dropped 代表封包由于有问题而遭丢弃的数量等等
- TX：与 RX 相反，为网络由启动到目前为止的传送情况；
- collisions：代表封包碰撞的情况，如果发生太多次， 表示您的网络状况不太好；
- txqueuelen：代表用来传输数据的缓冲区的储存长度；
- RX bytes, TX bytes：总传送、接收的字节总量
- Interrupt, Memory：网络卡硬件的数据， IRQ 岔断与内存地址；

透过观察上述的资料，大致上可以了解到您的网络情况，尤其是那个 RX, TX 内的 error 数量， 以及是否发生严重的 collision 情况，都是需要注意的喔！ ^_^

```
范例二：暂时修改网络接口
[root@linux ~]# ifconfig eth0 192.168.100.100
# 如果不加任何其它参数，则系统会依照该 IP 所在的 class 范围，
# 自动的计算出 netmask 以及 network, broadcast 等 IP 参数；

[root@linux ~]# ifconfig eth0 192.168.100.100 netmask 255.255.255.128 \
> mtu 8000
# 设定网络接口，同时设定 MTU 的数值！

[root@linux ~]# ifconfig eth0 mtu 9000
# 仅修改该接口的 MTU 数值，其它的保持不动！

[root@linux ~]# ifconfig eth0:0 192.168.50.50
# 仔细看那个界面， eth0:0 喔！那就是在该网络接口上，再仿真一个网络接口，
# 亦即是在一张网络卡上面设定多个 IP 的意思啦！

[root@linux ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0F:EA:A3:06:A2
          inet addr:192.168.10.100  Bcast:192.168.10.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3669 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2892 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:667547 (651.9 KiB)  TX bytes:584799 (571.0 KiB)
          Interrupt:209 Memory:fb000000-0

eth0:0    Link encap:Ethernet  HWaddr 00:0F:EA:A3:06:A2
          inet addr:192.168.200.2  Bcast:192.168.200.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
          Interrupt:209 Memory:fb000000-0
# 仔细看，是否与硬件有关的信息都相同啊！没错！因为是同一张网卡嘛！

[root@linux ~]# ifconfig eth0:0 down
# 关掉 eth0:0 这个界面。如果想要启动 eth1 ，并且不给予任何网络参数，
# ifconfig eth1 up 就可以达到了！

[root@linux ~]# /etc/init.d/network restart
# 刚刚设定的数据全部失效，会以 ifcfg-ethx 的设定为主！
```

呵呵！使用 ifconfig 可以暂时手动来设定或修改某个适配卡的相关功能， 并且也可以透过 eth0:0 这种虚拟的网络接口来设定好一张网络卡上面的多个 IP 喔！手动的方式真是简单啊！并且设定错误也不打紧，因为我们可以利用 /etc/init.d/network restart 来重新启动整个网络接口，那么之前手动的设定数据会全部都失效喔！另外， 要启动某个网络接口，但又不让他具有 IP 参数时，直接给他 ifconfig eth0 up 即可！ 这个动作经常在无线网卡当中会进行， 因为我们必须要启动无线网卡让他去侦测 AP 存在与否啊！

- ifup, ifdown

实时的手动修改一些网络接口参数，可以利用 ifconfig 来达成，如果是要直接以设定档， 亦即是在 /etc/sysconfig/network-scripts 里面的 ifcfg-ethx 等档案的设定参数来启动的话， 那就得要透过 ifdown 或 ifup 来达成了。

```
[root@linux ~]# ifup   {interface}
[root@linux ~]# ifdown {interface}

[root@linux ~]# ifup eth0
```

ifup 与 ifdown 真是太简单了！这两支程序其实是 script 而已，他会直接到 /etc/sysconfig/network-scripts 目录下搜寻对应的设定档，例如 ifup eth0 时，他会找出 ifcfg-eth0 这个档案的内容，然后来加以设定。 关于 ifcfg-eth0 的设定则请参考前一章连上 Internet 的说明。

不过，由于这两支程序主要是搜寻设定文件 (ifcfg-ethx) 来进行启动与关闭的， 所以在使用前请确定 ifcfg-ethx 是否真的存在于正确的目录内，否则会启动失败喔！ 另外，如果以 ifconfig eth0 .... 来设定或者是修改了网络接口后， 那就无法再以 ifdown eth0 的方式来关闭了！因为 ifdown 会分析比对目前的网络参数与 ifcfg-eth0 是否相符，不符的话，就会放弃该次动作。因此，使用 ifconfig 修改完毕后，应该要以 ifconfig eth0 down 才能够关闭该接口喔！

路由修改 route

我们在网络基础的时候谈过关于路由的问题， 两部主机之间一定要有路由才能够互通 TCP/IP 的协议，否则就无法进行联机啊！ 一般来说，只要有网络接口，该接口就会产生一个路由，例如在鸟哥实验室内部的主机有一个 eth0 及 lo ， 所以：

```
[root@linux ~]# route [-nee]
[root@linux ~]# route add [-net|-host] [网域或主机] netmask [mask] [gw|dev]
```

```
[root@linux ~]# route del [-net|-host] [网域或主机] netmask [mask] [gw|dev]
观察的参数：
   -n  ：不要使用通讯协议或主机名称，直接使用 IP 或 port number;
   -ee ：使用更详细的信息来显示
增加 (add) 与删除 (del) 路由的相关参数：
   -net    ：表示后面接的路由为一个网域;
   -host   ：表示后面接的为连接到单部主机的路由;
   netmask ：与网域有关，可以设定 netmask 决定网域的大小;
   gw      ：gateway 的简写，后续接的是 IP 的数值喔，与 dev 不同;
   dev     ：如果只是要指定由那一块网络卡联机出去，则使用这个设定，后面接 eth0 等

范例一：单纯的观察路由状态
[root@linux ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.10.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth0
0.0.0.0         192.168.10.30   0.0.0.0         UG    0      0        0 eth0

[root@linux ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.10.0    *               255.255.255.0   U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
default         server.cluster  0.0.0.0         UG    0      0        0 eth0
```

由上面的例子当中仔细观察 route 与 route -n 的输出结果，你可以发现有加 -n 参数的主要是显示出 IP ，至于使用 route 而已的话，显示的则是『主机名称』喔！ 也就是说，在预设的情况下， route 会去找出该 IP 的主机名称，如果找不到呢？ 就会显示的钝钝的(有点小慢)，所以说，鸟哥通常都直接使用 route -n 啦！ 由上面看起来，我们也知道 default = 0.0.0.0/0.0.0.0 ， 而上面的信息有哪些你必须要知道的呢？

- Destination, Genmask：这两个玩意儿就是分别是 network 与 netmask 啦！所以这两个咚咚就组合成为一个完整的网域啰！
- Gateway：该网域是通过那个 gateway 连接出去的？ 如果显示 0.0.0.0 表示该路由是直接由本机传送，亦即可以透过局域网络的 MAC 直接传讯； 如果有显示 IP 的话，表示该路由需要经过路由器 (通讯闸) 的帮忙才能够传送出去。
- Flags：总共有多个旗标，代表的意义如下：
  - U (route is up)：该路由是启动的;
  - H (target is a host)：目标是一部主机 (IP) 而非网域;
  - G (use gateway)：需要透过外部的主机 (gateway) 来转递封包;
  - R (reinstate route for dynamic routing)：使用动态路由时，恢复路由信息的旗标;
  - D (dynamically installed by daemon or redirect)：已经由服务或转 port 功能设定为动态路由
  - M (modified from routing daemon or redirect)：路由已经被修改了;

    - ! (reject route)：这个路由将不会被接受(用来抵挡不安全的网域！)
- Iface：这个路由传递封包的接口。

此外，观察一下上面的路由排列顺序喔，依序是由小网域 (192.168.10.0/24 是 Class C)，逐渐到大网域 (169.254.0.0/16 Class B) 最后则是预设路由 (0.0.0.0/0.0.0.0)。 然后当我们要判断某个网络封包应该如何传送的时候，该封包会经由这个路由的过程来判断喔！ 举例来说，我上头仅有三个路由，若我有一个传往 192.168.10.20 的封包要传递，那首先会找 192.168.10.0/24 这个网域的路由，找到了！所以直接由 eth0 传送出去；

如果是传送到 Yahoo 的主机呢？ Yahoo 的主机 IP 是 202.43.195.52，我通过判断 1)不是 192.168.10.0/24， 2)不是 169.254.0.0/16 结果到达 3)0/0 时，OK！传出去了，透过 eth0 将封包传给 192.168.10.30 那部 gateway 主机啊！所以说，路由是有顺序的。

因此当你重复设定多个同样的路由时， 例如在你的主机上的两张网络卡设定为相同网域的 IP 时，会出现什么情况？会出现如下的情况：

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.10.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.10.0    0.0.0.0         255.255.255.0   U     0      0        0 eth1
```

也就是说，由于路由是依照顺序来排列与传送的， 所以不论封包是由那个界面 (eth0, eth1) 所接收，都会由上述的 eth0 传送出去， 所以，在一部主机上面设定两个相同网域的 IP 本身没有什么意义！有点多此一举就是了。 除非是类似虚拟主机 (Xen, VMware 等软件) 所架设的多主机时，才会有这个必要～

```
范例二：路由的增加与删除
[root@linux ~]# route del -net 169.254.0.0 netmask 255.255.0.0 dev eth0
# 上面这个动作可以删除掉 169.254.0.0/16 这个网域！
# 请注意，在删除的时候，需要将路由表上面出现的信息都写入
# 包括  netmask , dev 等等参数喔！注意注意

[root@linux ~]# route add -net 192.168.100.0 \
> netmask 255.255.255.0 dev eth0
# 透过 route add 来增加一个路由！请注意，这个路由必须要能够与你互通。
# 举例来说，如果我下达底下的指令就会显示错误：
# route add -net 192.168.200.0 netmask 255.255.255.0 gw 192.168.200.254
# 因为我的环境内仅有 192.168.10.100 这个 IP ，所以不能与 192.168.200.254
# 这个网段直接使用 MAC 互通！这样说，可以理解喔！？

[root@linux ~]# route add default gw 192.168.10.30
# 增加预设路由的方法！请注意，只要有一个预设路由就够了喔！
# 在这个地方如果您随便设定后，记得使用底下的指令重新设定你的网络
# /etc/init.d/network restart
```

如果是要进行路由的删除与增加，那就得要参考上面的例子了， 其实，使用 man route 里面的数据就很丰富了！仔细查阅一下啰！ 你只要记得，当出现『SIOCADDRT: Network is unreachable』 这个错误时，

肯定是由于 gw 后面接的 IP 无法直接与您的网域沟通 (Gateway 并不在你的网域内)， 所以，赶紧检查一下是否输入错误啊！加油吧！

---

ip

ip 是个指令喔！并不是那个 TCP/IP 的 IP 啦！这个 ip 指令的功能可多了！ 基本上，他就是整合了 ifconfig 与 route 这两个指令啰～不过， ip 可以达成的功能却又多更多！ 真是个相当厉害的指令。如果您有兴趣的话，请自行 vi /sbin/ifup ，就知道整个 ifup 就是利用 ip 这个指令来达成的。好了，如何使用呢？让我们来瞧一瞧先！

```
[root@linux ~]# ip [option] [动作] [指令]
参数：
option ：设定的参数，主要有：
    -s ：显示出该装置的统计数据(statistics)，例如总接受封包数等；
动作：亦即是可以针对哪些网络参数进行动作，包括有：
    link  ：关于装置 (device) 的相关设定，包括 MTU, MAC 地址等等
    addr/address ：关于额外的 IP 协议，例如多 IP 的达成等等；
    route ：与路由有关的相关设定
```

由上面的语法我们可以知道，ip 除了可以设定一些基本的网络参数之外，还能够进行额外的 IP 协议，包括多 IP 的达成，真是太完美了！底下我们就分三个部分 (link, addr, route) 来介绍这个 ip 指令吧！

- 关于装置接口 (device) 的相关设定： ip link

ip link 可以设定与装置 (device) 有关的相关设定，包括 MTU 以及该网络接口的 MAC 等等， 当然也可以启动 (up) 或关闭 (down) 某个网络接口啦！整个语法是这样的：

```
[root@linux ~]# ip [-s] link show  <== 单纯的查阅该装置相关的信息
[root@linux ~]# ip link set [device] [动作与参数]
参数：
show：仅显示出这个装置的相关内容，如果加上 -s 会显示更多统计数据；
set ：可以开始设定项目， device 指的是 eth0, eth1 等等界面代号；
动作与参数：包括有底下的这些动作：
   up|down  ：启动 (up) 或关闭 (down) 某个接口，其它参数使用预设的以太网络；
   address  ：如果这个装置可以更改 MAC 的话，用这个参数修改！
   name     ：给予这个装置一个特殊的名字；
   mtu      ：就是最大传输单元啊！

范例一：显示出所有的接口信息
[root@linux ~]# ip link show
1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:50:fc:22:9a:cb brd ff:ff:ff:ff:ff:ff
3: sit0: <NOARP> mtu 1480 qdisc noop
```

```
    link/sit 0.0.0.0 brd 0.0.0.0

[root@linux ~]# ip -s link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:50:fc:22:9a:cb brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    484011792  2247372  0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    2914104290 2867753  0       0       0       0
```

使用 ip link show 可以显示出整个装置接口的硬件相关信息，如上所示，包括网卡地址(MAC)、MTU 等等，比较有趣的应该是那个 sit0 的接口了，那个 sit0 的界面是用在 IPv4 及 IPv6 的封包转换上的， 对于我们仅使用 IPv4 的网络是没有作用的。 lo 及 sit0 都是主机内部所自行设定的。 而如果加上 -s 的参数后，则这个网络卡的相关统计信息就会被列出来， 包括接收 (RX) 及传送 (TX) 的封包数量等等，详细的内容与 ifconfig 所输出的结果相同的。

```
范例二：启动、关闭与设定装置的相关信息
[root@linux ~]# ip link set eth0 up
# 启动 eth0 这个装置接口；

[root@linux ~]# ip link set eth0 down
# 阿就关闭啊！简单的要命～

[root@linux ~]# ip link set eth0 mtu 1000
# 更改 MTU 的值，达到 1000 bytes，单位就是 bytes 啊！
```

更新网络卡的 MTU 使用 ifconfig 也可以达成啊！没啥了不起，不过，如果是要更改『网络卡代号、 MAC 地址的信息』的话，那可就得使用 ip 啰～不过，设定前得要先关闭该网络卡，否则会不成功。 如下所示：

```
范例三：修改网络卡代号、MAC 等参数
[root@linux ~]# ip link set eth0 name vbird
SIOCSIFNAME: Device or resource busy
# 因为该装置目前是启动的，所以不能这样做设定。你应该要这样做：

[root@linux ~]# ip link set eth0 down       <==关闭界面
[root@linux ~]# ip link set eth0 name vbird <==重新设定
[root@linux ~]# ip link show                <==观察一下
2. vbird: <BROADCAST,MILTICASE> mtu 900 qdisc pfifo_fast qlen 1000
    link/ehter 00:40:d0:13:c3:46 brd ff:ff:ff:ff:ff:ff
# 怕了吧！连网络卡代号都可以改变！不过，玩玩后记得改回来啊！
# 因为我们的 ifcfg-eth0 还是使用原本的装置代号！避免有问题，要改回来
[root@linux ~]# ip link set vbird name eth0 <==界面改回来

[root@linux ~]# ip link set eth0 address aa:aa:aa:aa:aa:aa
[root@linux ~]# ip link show eth0
# 如果你的网络卡支持硬件地址 (MAC) 可以更改的话，
```

```
# 那么上面这个动作就可以更改你的网络卡地址了！厉害吧！
# 不过，还是那句老话，测试完之后请立刻改回来啊！
```

在这个装置的硬件相关信息设定上面，包括 MTU, MAC 以及传输的模式等等，都可以在这里设定。 有趣的是那个 address 的项目，那个项目后面接的可是硬件地址 (MAC) 而不是 IP 喔！ 很容易搞错啊！切记切记！更多的硬件参数可以使用 man ip 查阅一下与 ip link 有关的设定。

- 关于额外的 IP 相关设定： ip address

如果说 ip link 是与 OSI 七层协定 的第二层资料连阶层有关的话，那么 ip address (ip addr) 就是与第三层网络层有关的参数啦！ 主要是在设定与 IP 有关的各项参数，包括 netmask, broadcast 等等。

```
[root@linux ~]# ip address show    <==就是查阅 IP 参数啊！
[root@linux ~]# ip address [add|del] [IP 参数] [dev 装置名] [相关参数]
参数：
show     : 单纯的显示出接口的 IP 信息啊；
add|del : 进行相关参数的增加 (add) 或删除 (del) 设定，主要有：
    IP 参数：主要就是网域的设定，例如 192.168.100.100/24 之类的设定喔；
    dev     : 这个 IP 参数所要设定的接口，例如 eth0, eth1 等等；
    相关参数：主要有底下这些：
        broadcast：设定广播地址，如果设定值是 + 表示『让系统自动计算』
        label    : 亦即是这个装置的别名，例如 eth0:0 就是了！
        scope    : 这个界面的领域，通常是这几个大类：
                   global : 允许来自所有来源的联机；
                   site   : 仅支持 IPv6 ，仅允许本主机的联机；
                   link   : 仅允许本装置自我联机；
                   host   : 仅允许本主机内部的联机；
                   所以当然是使用 global 啰！预设也是 global 啦！

范例一：显示出所有的接口之 IP 参数：
[root@linux ~]# ip address show
1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:50:fc:22:9a:cb brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.2/24 brd 192.168.1.255 scope global eth0
    inet6 fe80::250:fcff:fe22:9acb/64 scope link
       valid_lft forever preferred_lft forever
3: sit0: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0
```

看到上面那个特殊的字体吗？没错！那就是 IP 参数啦！也是 ip address 最主要的功能。 底下我们进一步来新增虚拟的网络介面试看看：

```
范例二：新增一个接口，名称假设为 eth0:vbird
[root@linux ~]# ip address add 192.168.50.50/24 broadcast + \
> dev eth0 label eth0:vbird
[root@linux ~]# ip address show eth0
2: eth0:  mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:40:d0:13:c3:46 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global eth0
    inet 192.168.50.50/24 brd 192.168.50.255 scope global eth0:vbird
    inet6 fe80::240:d0ff:fe13:c346/64 scope link
       valid_lft forever preferred_lft forever
# 看到上面的特殊字体了吧？多出了一行新的接口，且名称是 eth0:vbird
# 至于那个 broadcast + 也可以写成 broadcast 192.168.50.255 啦！
[root@linux ~]# ifconfig
eth0:vbir Link encap:Ethernet  HWaddr 00:40:D0:13:C3:46
          inet addr:192.168.50.50  Bcast:192.168.50.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:5 Base address:0x3e00
# 如果使用 ifconfig 就能够看到这个怪东西了！可爱吧！ ^_^

范例三：将刚刚的界面删除
[root@linux ~]# ip address del 192.168.50.50/24 dev eth0
# 删除就比较简单啊！ ^_^
```

- 关于路由的相关设定： ip route

呵呵，这个项目当然就是路由的观察与设定啰！事实上， ip route 的功能几乎与 route 这个指令差不多，但是，他还可以进行额外的参数设计，例如 MTU 的规划等等，相当的强悍啊！

```
[root@linux ~]# ip route show  <==单纯的显示出路由的设定而已
[root@linux ~]# ip route [add|del] [IP 或网域] [via gateway] [dev 装置]
参数：
show : 单纯的显示出路由表，也可以使用 list ;
add|del : 增加 (add) 或删除 (del) 路由的意思。
    IP 或网域：可使用 192.168.50.0/24 之类的网域或者是单纯的 IP ;
    via      : 从那个 gateway 出去，不一定需要；
    dev      : 由那个装置连出去，这就需要了！
    mtu      : 可以额外的设定 MTU 的数值喔！

范例一：显示出目前的路由资料
[root@linux ~]# ip route show
192.168.1.0/24 dev eth0  proto kernel  scope link  src 192.168.1.2
169.254.0.0/16 dev eth1  scope link
default via 192.168.1.254 dev eth1
```

如上表所示，最简单的功能就是显示出目前的路由信息，其实跟 route 这个指令相同啦！ 指示必须要注意几个小东西：

- proto：此路由的路由协议，主要有 redirect, kernel, boot, static, ra 等， 其中 kernel 指的是直接由核心判断自动设定。
- scope：路由的范围，主要是 link ，亦即是与本装置有关的直接联机。

再来看一下如何进行路由的增加与删除吧！

```
范例二：增加路由，主要是本机直接可沟通的网域
[root@linux ~]# ip route add 192.168.5.0/24 dev eth0
# 针对本机直接沟通的网域设定好路由，不需要透过外部的路由器
[root@linux ~]# ip route show
192.168.5.0/24 dev eth0  scope link
....以下省略....

范例三：增加可以通往外部的路由，需透过 router 喔！
[root@linux ~]# ip route add 192.168.10.0/24 via 192.168.5.100 dev eth0
[root@linux ~]# ip route show
192.168.5.0/24 dev eth0  scope link
....其它省略....
192.168.10.0/24 via 192.168.5.100 dev eth0
# 仔细看喔，因为我有 192.168.5.0/24 的路由存在 (我的网卡直接联系)，
# 所以才可以将 192.168.10.0/24 的路由丢给 192.168.5.100
# 那部主机来帮忙传递喔！与之前提到的 route 指令是一样的限制！

范例四：增加预设路由
[root@linux ~]# ip route add default via 192.168.1.2 dev eth0
# 那个 192.168.1.2 就是我的预设路由器 (gateway) 的意思啊！ ^_^
# 真的记得，只要一个预设路由就 OK ！

范例五：删除路由
[root@linux ~]# ip route del 192.168.10.0/24
[root@linux ~]# ip route del 192.168.5.0/24
```

事实上，这个 ip 的指令实在是太博大精深了！刚接触 Linux 网络的朋友，可能会看到有点晕～ 不要紧啦！您先会使用 ifconfig, ifup , ifdown 与 route 即可， 等以后有经验了之后，再继续回来玩 ip 这个好玩的指令吧！ ^_^ 有兴趣的话，也可以自行参考 ethtool 这个指令喔！ (man ethtool)。

---

iwlist, iwconfig

这两个指令您必须要有无线网卡才能够进行喔！这两个指令的用途是这样的：

- iwlist：利用无线网卡进行无线 AP 的侦测与取得相关的数据；

- iwconfig：设定无线网卡的相关参数。

这两个指令的应用我们在前一章里面的 无线网卡设定 谈了很多了，所以这里我们不再详谈，有兴趣的朋友应该先使用 man iwlist 与 man iwconfig 了解一下语法， 然后再到前一章的无线网络小节查一查相关的用法，就了解了啦！ ^_^

---

dhclient

如果你是使用 DHCP 协议在局域网络内取得 IP 的话，那么是否一定要去编辑 ifcfg-eth0 内的 BOOTPROTO 呢？ 嘿嘿！有个更快速的作法，那就是利用 dhclient 这个指令～因为这个指令才是真正发送 dhcp 要求工作的程序啊！那要如何使用呢？很简单！如果不考虑其它的参数，使用底下的方法即可：

```
[root@linux ~]# dhclient eth0
```

够简单吧！这样就可以立刻叫我们的网络卡以 dhcp 协议去尝试取得 IP 喔！ 不过在 SuSE distribution 里面，他仅有 dhcpcd 这支程序，他与 dhclient 是相同的咚咚啦！ ^_^

网络侦错与观察指令：

在网络的互助论坛中，最常听到的一句话就是：『高手求救！我的 Linux 不能连上网络了！』我的天呐！不能上网络的原因多的很！而要完全搞懂也不是一件简单的事情呢！ 不过，事实上我们可以自己使用测试软件来追踪可能的错误原因，而很多的网络侦测指令其实在 Linux 里头已经都预设存在了，只要您好好的学一学基本的侦测指令，那么一些朋友在告诉您如何侦错的时候， 您应该就立刻可以知道如何来搞定他啰！好了，底下我们就简单的来谈一谈几个很基本的网络常用的侦错指令啦！

---

ping

这个 ping 是很重要的指令，ping 主要透过 ICMP 封包 来进行整个网络的状况报告，当然啦，最重要的就是那个 ICMP type 0, 8 这两个类型， 分别是要求回报与主动回报网络状态是否存在的特性。要特别注意的是， ping 还是需要透过 IP 封包来传送 ICMP 封包的， 而 IP 封包里面有个相当重要的 TTL (Time To Live) 属性，这是很重要的一个路由特性， 详细的 IP 与 ICMP 表头资料请参考网络基础的详细介绍。

```
[root@linux ~]# ping [-bcstnM] IP
参数：
-b ：后面接的是 broadcast 的 IP，用在你『需要对整个网域的主机进行 ping 』时；
-c ：后面接的是执行 ping 的次数，例如 -c 5 ；
-n ：不进行 IP 与主机名称的反查，直接使用 IP ；
-s ：发送出去的 ICMP 封包大小，预设为 56(bytes)，再加 8 bytes 的 ICMP 表头资料
-t ：TTL 的数值，预设是 255，每经过一个节点就会少一；
-M [do|dont] ：主要在侦测网络的 MTU 数值大小，两个常见的项目是：
   do  ：代表传送一个 DF (Don't Fragment) 旗标，让封包不能重新拆包与打包；
   dont：代表不要传送 DF 旗标，表示封包可以在其它主机上拆包与打包
```

```
范例一：侦测一下 168.95.1.1 这部 DNS 主机是否存在？
[root@linux ~]# ping -c 3 168.95.1.1
PING 168.95.1.1 (168.95.1.1) 56(84) bytes of data.
64 bytes from 168.95.1.1: icmp_seq=0 ttl=243 time=9.16 ms
64 bytes from 168.95.1.1: icmp_seq=1 ttl=243 time=8.98 ms
64 bytes from 168.95.1.1: icmp_seq=2 ttl=243 time=8.80 ms

--- 168.95.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 8.807/8.986/9.163/0.164 ms, pipe 2
```

ping 最简单的功能就是传送 ICMP 封包去要求对方主机响应是否存在于网络环境中， 上面的响应讯息当中，几个重要的项目是这样的：

- 64 bytes：表示这次传送的 ICMP 封包大小为 64 bytes 这么大，这是默认值， 在某些特殊场合中，例如要搜索整个网络内最大的 MTU 时，可以使用 -s 2000 之类的数值来取代；
- icmp_seq=0：ICMP 所侦测进行的次数，第一次编号为 0 ；
- ttl=243：TTL 与 IP 封包内的 TTL 是相同的，每经过一个带有 MAC 的节点 (node) 时，例如 router, bridge 时， TTL 就会减少一，预设的 TTL 为 255 ， 你可以透过 -t 150 之类的方法来重新设定预设 TTL 数值；
- time=9.16 ms：响应时间，单位有 ms(0.001 秒)及 us(0.000001 秒)， 一般来说，越小的响应时间，表示两部主机之间的网络联机越良好！

如果你忘记加上 -c 3 这样的规定侦测次数，那就得要使用 [ctrl]-c 将他结束掉了！

```
范例二：针对整个网域进行 ping 的追查
[root@linux ~]# ping -c 3 -b 192.168.10.255
WARNING: pinging broadcast address          <==会告知危险喔！
PING 192.168.10.255 (192.168.10.255) 56(84) bytes of data.
64 bytes from 192.168.10.100: icmp_seq=1 ttl=64 time=0.177 ms
64 bytes from 192.168.10.20: icmp_seq=1 ttl=64 time=0.179 ms (DUP!)
64 bytes from 192.168.10.10: icmp_seq=1 ttl=64 time=0.302 ms (DUP!)
64 bytes from 192.168.10.40: icmp_seq=1 ttl=64 time=0.304 ms (DUP!)
# 当要针对整部主机作 ping 的侦测时，可以利用 -b 这个参数。
# 请特别注意，当使用 ping -b 时，会对整个网域进行侦测喔！没事别乱用。
# 例如上面的范例中，区网内的 192.168.10.20... 等主机会被侦测到。
```

如果想要了解区网内有多少部主机存活着，那么使用 ping -b broadcast 就能够知道了！ 而不必一部一部主机来侦测啊！方便～另外也特别注意一下， 如果您的主机与待侦测主机并不在同一个网域内， 那么 TTL 预设使用 255 ，如果是同一个网域内，那么 TTL 预设则使用 64 喔！ 看看上面的输出即可了解。

用 ping 追踪最大 MTU 数值

我们由前几章的网络基础里面谈到加大讯框 (frame) 时， 对于网络效能是有帮助的，因为封包打包的次数会减少，加上如果整个传输的媒体都能够接受这个 frame 而不需要重新进行封包的拆解与重组的话，那么效能当然会更好，那个修改 frame 大小的参数就是 MTU 啦！好了，现在我们知道网络卡的 MTU 可以透

过 ifconfig 或者是 ip 等来达成， 那么追踪整个网络传输的最大 MTU 时，又该如何查询？呵呵！最简单的方法当然是透过 ping 传送一个大封包， 并且不许中继的路由器或 switch 将该封包重组，那就能够处理啦！没错！可以这样的：

```
范例三：找出最大的 MTU 数值
[root@linux ~]# ping -c 2 -s 1000 -M do 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 1000(1028) bytes of data.
1008 bytes from 192.168.10.10: icmp_seq=1 ttl=64 time=0.424 ms
# 如果有响应，那就是可以接受这个封包，如果无响应，那就表示这个 MTU 太大了。

[root@linux ~]# ping -c 2 -s 8000 -M do 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 8000(8028) bytes of data.
ping: local error: Message too long, mtu=1500
# 这个错误讯息是说，本地端的 MTU 才到 1500 而已，你要侦测 8000 的 MTU
# 根本就是无法达成的！那要如何是好？用前一小节介绍的 ip link 来进行 MTU 设定吧！
```

不过，你需要知道的是，由于 IP 封包表头 (不含 options) 就已经占用了 20 bytes ，再加上 ICMP 的表头有 8 bytes ，所以当然你在使用 -s size 的时候，那个封包的大小就得要先扣除 (20+8=28) 的大小了。 因此如果要使用 MTU 为 1500 时，就得要下达『 ping -s 1472 -M do xx.yy.zz.ip 』才行啊！ 另外，由于本地端的网络卡 MTU 也会影响到侦测，所以如果想要侦测整个传输媒体的 MTU 数值， 那么每个可以调整的主机就得要先使用 ifcofig 或 ip 先将 MTU 调大，然后再去进行侦测， 否则就会出现像上面提供的案例一样，可能会出现『Message too long, mtu=1500』之类的字样喔！ 至于如果侦测完毕后，想要调整最佳化的 MTU ，那么请参考前一章节的内容来调整啰！ ^_^

不过这个 MTU 不要随便调整啊！除非真的有问题。通常调整 MTU 的时间是在这个时候：

- 因为全部的主机群都是在内部的区网，例如丛集架构 (cluster) 的环境下， 由于内部的网络节点都是我们可以控制的，因此可以透过修改 MTU 来增进网络效能；
- 因为操作系统预设的 MTU 与您的网域不符，导致某些网站可以顺利联机，某些网站则无法联机。以 Windows 操作系统作为联机分享的主机时，在 Client 端挺容易发生这个问题；

如果是要连上 Internet 的主机，注意不要随便调整 MTU ，因为我们无法知道 Internet 上面的每部机器能够支持的 MTU 到多大，因为......不是我们能够管的到的嘛！ ^_^

另外，其实每种联机方式都有不同的 MTU 值，常见的各种接口的 MTU 值分别为：

网络接口	MTU
Ethernet	1500
PPPoE	1492
Dial-up(Modem)	576

网络上也有免费帮忙查询 MTU 与传输相关数据的网站，例如底下这个网站：

- http://forums.speedguide.net:8117/

连接上这个网站之前，请先取消您浏览器上的代理服务器 (Proxy) 的设定，才能显示出正确的讯息。 如果在 Windows 的系统上面想要修改 MTU 值的话，那就得要修改 Windows 的登录档， 在 Windows 上面对于 MTU 的侦测与修改的详细作法可以参考微软的官方网站：

- http://www.microsoft.com/taiwan/msclub/member/TIPS/Spring_2001/tip1to3/tip1to3_2.htm

---

traceroute

我们前面谈到的指令大多数都是针对主机的网络参数设定所需要的，而 ping 是两部主机之间的回声与否判断， 那么有没有指令可以追踪两部主机之间通过的各个节点 (node) 通讯状况的好坏呢？ 举例来说，如果我们联机到 yahoo 的速度比平常慢，你觉得是 (1)自己的网络环境有问题？ (2)还是外部的 Internet 有问题？如果是 (1) 的话，我们当然需要检查自己的网络环境啊， 看看是否又有谁中毒了？但如果是 Internet 的问题呢？那只有『等等等』啊！ 判断是 (1) 还是 (2) 就得要使用 traceroute 这个指令啦！

```
[root@linux ~]# traceroute [-nwig] IP
参数：
-n ：可以不必进行主机的名称解析，单纯用 IP ，速度较快！
-w ：若对方主机在几秒钟内没有回声就宣告不治...预设是 5 秒
-i ：用在比较复杂的环境，如果你的网络接口很多很复杂时，才会用到这个参数；
     举例来说，你有两条 ADSL 可以连接到外部，那你的主机会有两个 ppp，
     你可以使用 -i 来选择是 ppp0 还是 ppp1 啦！
-g ：与 -i 的参数相仿，只是 -g 后面接的是 gateway 的 IP 就是了。

范例一：
[root@linux ~]# traceroute -n tw.yahoo.com
traceroute to tw.yahoo-ap1.akadns.net (203.84.202.164), 30 hops max, 38 byte packets
 1  61.59.121.1  42.174 ms  41.690 ms  41.058 ms
 2  139.175.172.2  40.962 ms  41.978 ms  40.973 ms
 3  192.72.122.130  40.983 ms  41.930 ms  41.003 ms
 4  139.175.58.210  42.956 ms  41.997 ms  42.337 ms
 5  139.175.58.153  47.591 ms  47.972 ms  48.748 ms
 6  139.175.56.30  48.193 ms  47.970 ms  47.986 ms
 7  139.175.57.94  47.959 ms  47.951 ms  47.985 ms
 8  139.175.56.138  48.363 ms  47.586 ms  47.995 ms
 9  139.175.58.42  49.256 ms  50.668 ms  47.490 ms
10  61.58.33.133  201.882 ms  201.565 ms  200.973 ms
11  61.58.33.50  199.910 ms  199.019 ms  198.961 ms
12  203.84.200.226  202.391 ms  202.567 ms  209.283 ms
```

这个 traceroute 挺有意思的，这个指令会针对欲连接的目的地之所有 router 进行 ICMP 的逾时等待，例如上面的例子当中，由鸟哥的主机连接到 Yahoo 时，他会经过 12 个节点，traceroute 会主动的对这 12 个节点做 ICMP 的回声等待，并侦测回复的时间，每个节点会侦测三次。 所以像上头显示的结果，发现每个节点其实回复的时间大约在 200 ms 以内，算是还可以的 Internet 环境了。 而且由上面的信息来看，可以看出在 61.58.33.133 这个节点后的传输延迟较久，至于之前的 9 个节点则有不错的表现。透过这种

解析，可以让您了解到这条联机是那个环节出了问题喔。

另外，如果在预设的 5 秒钟之内 traceroute 听不到节点的回声，那么屏幕上就会跑出一个『 * 』的符号， 告知该节点无法有顺利的响应。由于我们的 traceroute 用的是 ICMP 封包，有些防火墙或者主机可能会将 ICMP 可通过的权力拿掉，因此就会造成等不到回声的状态！另外，有些 gateway 本来就不支持 traceroute 的功能，因此也会产生那个『 * 』的状况。所以分析时得要注意一下呐！

---

netstat

如果你觉得你的某个网络服务明明就启动了，但是就是无法造成联机的话，那么应该怎么办？ 首先你应该要查询一下自己的网络接口所监听的端口口 (port) 来看看是否真的有启动， 因为有时候屏幕上面显示的 [OK] 并不一定是 OK 啊！ ^_^

```
[root@linux ~]# netstat -[rn]       <==与路由有关的参数
[root@linux ~]# netstat -[antulpc]  <==与网络接口有关的参数
参数：
与路由 (route) 有关的参数说明：
-r  ：列出路由表(route table)，功能如同 route 这个指令；
-n  ：不使用主机名称与服务名称，使用 IP 与 port number ，如同 route -n
与网络接口有关的参数：
-a  ：列出所有的联机状态，包括 tcp/udp/unix socket 等；
-t  ：仅列出 TCP 封包的联机；
-u  ：仅列出 UDP 封包的联机；
-l  ：仅列出有在 Listen (监听) 的服务之网络状态；
-p  ：列出 PID 与 Program 的檔名；
-c  ：可以设定几秒钟后自动更新一次，例如 -c 5 每五秒更新一次网络状态的显示；

范例一：列出目前的路由表状态，且以 IP 及 port number 显示：
[root@linux ~]# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.10.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 eth0
0.0.0.0         192.168.10.30   0.0.0.0         UG        0 0          0 eth0
# 其实这个参数就跟 route -n 一模一样，对吧！这不是 netstat 的主要功能啦！

范例二：列出目前的所有网络联机状态，使用 IP 与 port number
[root@linux ~]# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:25                  0.0.0.0:*                   LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
tcp        0      0 ::ffff:192.168.10.100:25    ::ffff:192.168.10.200:57509 TIME_WAIT
tcp        0     52 ::ffff:192.168.10.100:22    ::ffff:192.168.10.210:1504  ESTABLISHED
```

```
udp        0      0 127.0.0.1:53                0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     4792   public/cleanup
unix  2      [ ACC ]     STREAM     LISTENING     4799   private/rewrite
......(底下省略)......
```

netstat 的输出主要分为两大部分，分别是 TCP/IP 的网络接口部分，以及传统的 Unix socket 部分。 还记得我们在基础篇里面曾经谈到档案的类型吗？那个 socket 与 FIFO 档案还记得吧？ 那就是在 Unix 接口用来做为程序数据交流的接口了，也就是上头表格内看到的 Active Unix domain sockets 的内容啰～

通常鸟哥都是建议加上『 -n 』这个参数的，因为可以避过主机名称与服务名称的反查，直接以 IP 及端口口号码 (port number) 来显示，显示的速度上会快很多！至于在输出的讯息当中， 我们先来谈一谈关于网络联机状态的输出部分，他主要是分为底下几个大项：

- Proto：该联机的封包协议，主要为 TCP/UDP 等封包；
- Recv-Q：非由使用者程序连接所复制而来的总 bytes 数；
- Send-Q：由远程主机所传送而来，但不具有 ACK 标志的总 bytes 数， 意指主动联机 SYN 或其它标志的封包所占的 bytes 数；
- Local Address：本地端的地址，可以是 IP (-n 参数存在时)， 也可以是完整的主机名称。如上表我们看到的 IP 格式有两种，一种是 IPv4 的标准，亦即是四组十进制的数字后面加上冒号『:』后，接着 port number 。一种是 IPv6 ， 前面的 IP 加上很多冒号『:』的格式。我们可以由这个显示的数据看出这个服务是开放在哪一个接口， 例如上表当中， port 22 是开放在 0.0.0.0 ，亦即是所有接口都可以连到 port 22 ， 至于 port 53 则仅开放在本机的 127.0.0.1 这个接口而已，所以是不对外部接口开放的意思。
- Foreign Address：远程的主机 IP 与 port number
- stat：状态列，主要的状态含有：
    - ESTABLISED：已建立联机的状态；
    - SYN_SENT：发出主动联机 (SYN 标志) 的联机封包；
    - SYN_RECV：接收到一个要求联机的主动联机封包；
    - FIN_WAIT1：该插槽服务(socket)已中断，该联机正在断线当中；
    - FIN_WAIT2：该联机已挂断，但正在等待对方主机响应断线确认的封包；
    - TIME_WAIT：该联机已挂断，但 socket 还在网络上等待结束；
    - LISTEN：通常用在服务的监听 port ！可使用『 -l 』参数查阅。

基本上，我们常常谈到的 netstat 的功能，就是在观察网络的联机状态了，而网络联机状态中， 又以观察『我目前开了多少的 port 在等待客户端的联机』以及 『目前我的网络联机状态中，有多少联机已建立或产生问题』最常见。 那你如何了解与观察呢？通常鸟哥是这样处理的：

```
范例三：秀出目前已经启动的网络服务
[root@linux ~]# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address     Foreign Address   State    PID/Program name
tcp        0      0 0.0.0.0:25        0.0.0.0:*         LISTEN   2141/master
tcp        0      0 :::22             :::*              LISTEN   1924/sshd
```

```
tcp        0      0 :::25           :::*            LISTEN    2141/master
udp        0      0 127.0.0.1:53    0.0.0.0:*                 1911/named
# 上面最重要的其实是那个 -l 的参数，因为可以仅列出有在 Listen 的 port
```

你可以在上面的范例当中发现，我的网络联机仅有对外开放 port 25 以及 port 22 而已 (因为针对 0.0.0.0 开放)， 至于 port 53 则仅针对内部的 127.0.0.1 来开放，所以是不对 Internet 开放这个服务的喔！ 而其中 port 22,25 都是使用 TCP 封包，至于 port 53 则是开放在 UDP 封包的状态！ 再仔细的看，每一行输出的最右边，你可以发现鸟哥的主机 port 22 是由 sshd 这支程序所启动的， 并且他的 PID 是 1924 ，看到这边，聪明的您应该知道，『那我如何关闭这个 port 』吧？使用 kill 或 killall 即可啊！ ^_^

```
范例四：观察本机上头所有的网络联机状态
[root@linux ~]# netstat -atunp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address      State       PID/Program name
tcp        0      0 0.0.0.0:25          0.0.0.0:*            LISTEN      2141/master
tcp        0      0 :::22               :::*                 LISTEN      1924/sshd
tcp        0      0 :::25               :::*                 LISTEN      2141/master
tcp        0     68 192.168.1.100:22    192.168.1.210:1504   ESTABLISHED 30417/sshd:
udp        0      0 127.0.0.1:53        0.0.0.0:*                        1911/named
```

看到上头的特殊字体吧？那代表目前已经建立联机的一条网络联机，他是由远程主机 192.168.1.210 启动一个大于 1024 的埠口向本地端主机 192.168.1.100 的 port 22 进行联机的一条联机， 你必须要想起来的是：『Client 端是随机取一个大于 1024 以上的 port 进行联机』，此外『只有 root 可以启动小于 1023 以下的 port 』，那就看的懂上头那条联机啰！如果这条联机你想要砍掉他的话， 看到最右边的 30417/sshd 了没？ kill 会用吧！ ^_^

至于传统的 Unix socket 的数据，记得使用 man netstat 查阅一下吧！ 这个 Unix socket 通常是用在一些仅在本机上运作的程序所开启的插槽接口文件， 例如 X Window 不都是在本机上运作而已吗？那何必启动网络的 port 呢？当然可以使用 Unix socket 啰，另外，例如 Postfix 这一类的网络服务器，由于很多动作都是在本机上头来完成的， 所以以会占用很多的 Unix socket 喔！

例题一：请说明服务名称与 port number 的对应在 Linux 当中，是用那个档案来设定对应的？

答：

/etc/services

---

host

这个指令可以用来查出某个主机名称的 IP 喔！举例来说，我们想要知道 tw.yahoo.com 的 IP 时， 可以这样做：

```
[root@linux ~]# host [-a] hostname [server]
参数：
```

```
-a ：列出该主机详细的各项主机名称设定数据
[server] ：可以使用非为 /etc/resolv.conf 的 DNS 主机来查询。

范例一：列出 tw.yahoo.com 的 IP
[root@linux ~]# host tw.yahoo.com
tw.yahoo.com is an alias for tw.yahoo-ap1.akadns.net.
tw.yahoo-ap1.akadns.net has address 202.43.195.52
```

瞧！IP 是 202.43.195.52 啊！很简单就可以查询到 IP 了！ 那么这个 IP 是向谁查询的呢？其实就是写在 /etc/resolv.conf 那个档案内的 DNS 主机啦!如果不想要使用该档案内的主机来查询,也可以这样做:

```
[root@linux ~]# host tw.yahoo.com 168.95.1.1
Using domain server:
Name: 168.95.1.1
Address: 168.95.1.1#53
Aliases:

tw.yahoo.com is an alias for tw.yahoo-ap1.akadns.net.
tw.yahoo-ap1.akadns.net has address 202.43.195.52
```

会告诉我们所使用来查询的主机是哪一部呐!这样就够清楚了吧？至于更详细的 host 用法,我们会在 DNS 主机 那个章节再来好好聊一聊吧！

---

nslookup

这玩意儿的用途与 host 基本上是一样的，就是用来作为 IP 与主机名称对应的检查， 同样是使用 /etc/resolv.conf 这个档案来作为 DNS 服务器的来源选择。

```
[root@linux ~]# nslookup [-query=[type]] [hostname|IP]
参数：
-query=type：查询的类型，除了传统的 IP 与主机名称对应外，DNS 还有很多信息，
             所以我们可以查询很多不同的信息，包括 mx， cname 等等，
             例如： -query=mx 的查询方法！

范例一：找出 www.google.com.tw 的 IP
[root@linux ~]# nslookup www.google.com.tw
Server:         168.95.1.1
Address:        168.95.1.1#53

Non-authoritative answer:
www.google.com.tw       canonical name = www.google.com.
www.google.com  canonical name = www.l.google.com.
Name:   www.l.google.com
Address: 64.233.189.104
```

```
范例二：找出 168.95.1.1 的主机名称
[root@linux ~]# nslookup 168.95.1.1
Server:         168.95.1.1
Address:        168.95.1.1#53

1.1.95.168.in-addr.arpa name = dns.hinet.net.
```

如何，看起来与 host 差不多吧！不过，这个 nslookup 还可以由 IP 找出主机名称喔！ 例如那个范例二，他的主机名称是： dns.hinet.net 哩！目前大家都建议使用 dig 这个指令来取代 nslookup ，我们会在 DNS 服务器 那时再来好好谈一谈吧！

远程联机指令：

啥是远程联机呢？其实就是在不同的计算机之间进行登入的情况啦！ 我们可以透过 telnet, ssh 或者是 ftp 等协议来进行远程主机的登入， 底下我们就分别来介绍一下这些基本的指令吧！另外，相关的服务器我们则会在后续进行说明的。

---

telnet

telnet 是早期我们在个人计算机上面要连结到服务器主机上工作时，最重要的一个软件了！ 他不但可以直接连接到服务器上头，还可以用来连结 BBS 呢！非常棒！ 不过， telnet 本身的数据在传送的时候是使用明码（原始的数据，没有加密）， 所以数据在 Internet 上面跑的时候，会比较危险一点（就怕被别人监听啊）。 更详细的资料我们会在『远程联机服务器』章节内做介绍的。

```
[root@linux ~]# telnet [host|IP] [port]

范例一：连结到成大梦之大地这个 BBS 站
[root@linux ~]# telnet bbs.dorm.ncku.edu.tw
bbs.ccns.ncku.edu.tw ⊙ 梦之大地 逼逼乀四 ⊙ 140.116.250.3 [DreamBBS Ver.040223]
欢迎光临【 梦之大地 逼逼乀四 】。系统负载：0.16 0.16 0.16 [负载正常]
                                        ┌┤ 梦之大地 ├──────────┐
                                        │                      │
                                        │ 梦之大地由            │
                                        │   【计算机网络爱好社·CCNS】 │
                                        │             维护管理  │
                                        │                      │
                                        └────────────┤ By BenHe ├┘

  ┌┤ 本站站长群 ├──────────┐
  │站长: billcho           │
  │系统: cat               │
  │站务: muwell  ianwolf   │
  │      renn999 GG        │
```

```
参观用账号：guest，申请新账号：new。目前在线人数 [2183/5000] 人。
请输入代号：
```

如上所示，我们可以透过 telnet 轻易的连结到 BBS 上面，而如果您的主机有开启 telnet 服务的话，同样的利用『 telnet IP 』并且输入账号与密码之后，就能够登入主机了。 另外，在 Linux 上的 telnet 软件还提供了 Kerberos 的认证方式，有兴趣的话请自行参阅 man telnet 的说明。

除了连结到服务器以及连结到 BBS 站之外， telnet 还可以用来连结到某个 port (服务) 上头呐！ 举例来说，我们可以用 telnet 连接到 port 110 ，看看这个 port 是否有正确的启动呢？

```
范例二：侦测本机端的 110 这个 port 是否正确启动？
[root@linux ~]# telnet localhost 110
Trying 127.0.0.1...
telnet: connect to address 127.0.0.1: Connection refused
# 如果出现这样的讯息，代表这个 port 没有启动或者是这个联机有问题，
# 因为您看到那个 refused 嘛！

[root@linux ~]# telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 vbird.vbird.idv.tw ESMTP Postfix
ehlo localhost
250-linux.dm.tsai
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250 8BITMIME
quit
221 Bye
Connection closed by foreign host.
```

瞧！根据输出的结果，我们就能够知道这个通讯协议 (port number 提供的通讯协议功能) 是否有成功的启动呐！ 而在每个 port 所监听的服务都有其特殊的指令，例如上述的 port 25 就是在本机接口所提供的电子邮件服务， 那个服务所支持的指令就如同上面使用的数据一样，但是其它的 port 就不见得支持这个『 ehlo 』的命令， 因为不同的 port 有不同的程序嘛！所以当然支持的命令就不同啰！ 与 mail server 有关的 telnet 用法，我们将在邮件服务器内提到喔！

---

ftp

常常会听到『FTP』这个咚咚吧！举例来说，如果你想要下载 Linux 的光盘烧录映象文件时，要去哪里下载啊？ 不是说要去义守大学吗？也可以到成大或昆山科大等等的 FTP 网站，嘿嘿！没错～那就是 FTP 提

供者啦！那我们要如何去下载呢？当然就是透过 ftp 的客户端软件了。在 Linux 底下，我们可以透过 ftp 这个软件， 也可以透过下一小节会提到的 lftp 说～

```
[root@linux ~]# ftp [-p] [host|IP] [port]
参数：
-p ：启动被动式模式 (passive, PASV)；

范例一：联机到义守大学去看看
[root@linux ~]# ftp ftp.isu.edu.tw
Connected to ftp.isu.edu.tw (140.127.177.17).
220-欢迎光临义守大学档案服务器
220-
220-本站提供以下软件可供下载：
220-*******************************************************************************
220-/pub/BeOS/       BeOS 操作系统
220-/pub/Linux/      Linux 操作系统
....(其它省略)....
220-*******************************************************************************
Name (ftp.isu.edu.tw:dmtsai): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> █
ftp> help          <==提供更多的可用指令，可以常参考！
ftp> cd /pub       <==变换目录到 /pub 当中
ftp> dir           <==显示远程主机的目录内容
ftp> get file      <==下载 file 这个档案
ftp> mget file     <==下载 file 这个目录或档案
ftp> put file      <==上传 file 这个档案到服务器上
ftp> delete file   <==删除主机上的 file 这个档案
ftp> mkdir dir     <==建立 dir 这个目录
ftp> lcd /home     <==切换『本地端主机』的工作目录
ftp> passive       <==启动或关闭 passive 模式
ftp> binary        <==数据传输模式设定为 binary 格式
```

FTP 其实算是一个很麻烦的协议，因为他使用两个 port 分别进行命令与数据的交流， 详细的数据我们会在后续的 FTP 服务器内详谈，这里我们先单纯的介绍一下如何使用 ftp 这个软件。 首先我们当然是需要登入啰，所以在上头的表格当中我们当然需要填入账号与密码了。 不过由于义守大学提供匿名登入，而匿名登入者的账号就是『 anonymous 』所以直接填写那个账号即可。 如果是私人的 FTP 时，才需要提供一组完整的账号与密码啦！

登入 FTP 主机后，就能够使用 ftp 软件的功能进行上传与下载的动作， 几个常用的 ftp 内指令如上表，不过，鸟哥建议您可以连到大学的 FTP 网站后， 使用 help (或问号 ?) 来参考可用的指令，然后尝试下载以测试使用一下这个指令吧！ 这样以后没有浏览器的时候，你也可以到 ftp 下载了呢！不错吧！ ^_^

另外，如果由于某些理由，让你的 FTP 主机的 port 开在非正规的埠口时， 那你就可以利用底下的方式来连接到该部主机喔!

```
[root@linux ~]# ftp hostname 318
# 假设对方主机的 ftp 服务开启在 318 这个 port 啊!
```

---

lftp

早期当我们要登入提供匿名登入的主机时，很多时候都是使用 ncftp 这个软件， 不过，现在有更棒的选择，那就是 lftp 啦! 这个软件甚至可以在 ftp 里面使用类似 bash 的指令功能， 实在是非常的完美! 而整个使用的方法与上面提到的 ftp 又非常类似呐!

```
[root@linux ~]# lftp [-p port] [-u user[,pass]] [host|IP]
参数:
-p  : 后面可以直接接上远程 FTP 主机提供的 port
-u  : 后面则是接上 账号与密码 ，就能够连接上远程主机了
      如果没有加账号密码， lftp 预设会使用 anonymous 尝试匿名登入

范例一：利用 lftp 登入义守大学
[root@linux ~]# lftp ftp.isu.edu.tw
lftp ftp.isu.edu.tw:~> 
# 瞧! 一下子就登入了! 很快乐吧! ^_^
```

至于登入 FTP 主机后，一样可以使用『help』来显示出可以执行的指令， 与 ftp 很类似啦! 不过多了书签的功能，而且也非常的类似 bash 那! 很不错呦! 除了这个好用的文字接口的 FTP 软件之外，事实上还有很多图形接口的好用软件呢! 最常见的就是 gftp 了～不但是图形接口，而且与 cute ftp 简直就是像到不行! 非常的容易上手喔! CentOS 本身就有提供 gftp 了，你可以拿出原版的光盘来安装，然后进入 X Window 后， 启动一个 shell ，输入『 gftp 』就能够发现他的好用啦! 底下我们在来介绍一下实时通讯吧!

---

gaim

我想，现在应该大家都知道什么是 MSN, 雅虎实时通以及其它的通讯软件吧? 那么要连上这些服务器时，该怎么处理哪? 很简单,在 X Window 底下使用 gaim 就好了! 简直简单到不行～ ^_^ 请先进入 X Window 系统，然后开启一个终端机窗口，接着直接输入 gaim (请注意您必须已经安装了 gaim 了) 然后就会出现如下的窗口啦:



图一、gaim 使用范例图

在输入你的账号与密码，并选择相对应的实时通讯服务器（如 MSN 或 Yahoo 实时通），就可以进入到如下画面：



图二、gaim 使用范例图

若一切都没有问题后，按下『登入』嘿嘿～您就可以在 Linux 上头使用实时通讯软件啦！方便的很哩！ ^_^

文字接口网页浏览

什么？文字界面竟然有浏览器！别逗了好不好？呵呵！谁有那个时间在逗您呦！真的啦！有这个东西， 是在文字界面下上网浏览的好工具！分别是 lynx 及 wget 这两个宝贝蛋，但是，您必需要确定您已经安装了这两个套件才行。 底下就让我们来聊一聊这两个好用的家伙吧！

---

lynx

这个指令可以让我们来浏览网页，但鸟哥认为，这个档案最大的功能是在『 查阅 Linux 本机上面以 HTML 语法写成的文件数据 (document)』 怎么说呢？如果你曾经到 Linux 本机底下的 /usr/share/doc 这个目录看过文件数据的话， 就会常常发现一些网页档案，使用 vi 去查阅时，老是看到一堆 HTML 的语法！有碍阅读啊～ 这时候使用 lynx 就是个好方法啦！可以看的清清楚楚啊！ ^_^

```
[root@linux ~]# lynx [options] [website]
参数：
options 指的是一些惯用的参数，可以使用 man lynx 查阅，常见的有：
-anonymous ：预设使用匿名登入；
-assume_charset=big5 ：设定预设的语系数据为 big5 ，用在中文网页很方便

范例一：浏览 Linux kernel 网站
[root@linux ~]# LANG=zh_TW.big5
[root@linux ~]# lynx http://www.kernel.org
```

输入 LANG=zh_TW.big5 是当您想要浏览中文网站时，那么终端机就得要有相对应的显示编码才行， 否则会有一堆乱码产生啊！当我直接输入 lynx 网站网址后，就会出现如下的图示：



图三、lynx 使用范例图

在特殊字体的部分是我们可以使用 <tab> 按键来进行『超级链接』的按钮啦～而上图最底下一行则显示出一些热键， 你可以按上述的热键来参考一些常见的指令功能。不过有些地方您还是得要知道才行：

- 进入画面之后，由于是文字型态，所以编排可能会有点位移！不过不打紧！不会影响我们看咚咚！
- 这个时候可以使用『上下键』来让光标在上面的选项当中(如信箱、书签等等的)，按下 Enter 就进入该页面
- 可以使用『左右键』来移动『上一页或下一页』
- 可以藉由修改 /etc/lynx.cfg 来设定显示的字符编码 (台湾地区可以选择 Big5 编码)
- 其它的设定可以使用上面的范例当中，最底下那一行的说明喔！
- 一些常见功能：
    - h：Help, 求助功能. 在线说明书.
    - g：Goto URL, 按 g 后输入网页地址(URL) 如:http://www.abc.edu/等
    - d：download, 下载档案.
    - q：Quit, 跳离 lynx !
    - Ctrl+C ：强迫切断 lynx 的执行.
    - 方向键:
        - 上 ：移动光标至本页中 "上一个可连结点" .
        - 下 ：移动光标至本页中 "下一个可连结点" .
        - 左 ：back. 跳回上一页.
        - 右 ：进入反白光标所连结之网页.
        - ENTER 同 "右" 键.

至于如果是浏览 Linux 本机上面的网页档案，那就可以使用如下的方式：

```
[root@linux ~]# cd /usr/share/doc/samba-3.0.10/htmldocs
```

```
[root@linux htmldocs]# lynx index.html
```

在鸟哥的 CentOS 4.3 当中，有这么一个档案，我就可以利用 lynx 来取出察看呐！ 显示的结果有点像底下这样：



图四、lynx 使用范例图

当然啦！因为您的环境可能是在 Linux 本机的 tty1~tty6 ，所以无法显示出中文， 这个时候你就得要设定为：『LANG=en_US』之类的语系设定才行喔！ 而如果你常常需要浏览中文语系的网页，那就可以直接修改设定档，例如 /etc/lynx.cfg 这个档案内：

```
[root@linux ~]# vi /etc/lynx.cfg
CHARACTER_SET:utf-8              <==约在 399 行
#ASSUME_CHARSET:iso-8859-1       <==约在 414 行
#PREFERRED_LANGUAGE:en           <==约在 542 行

# 你可以将他改成如下所示：
CHARACTER_SET:big5
ASSUME_CHARSET:big5
PREFERRED_LANGUAGE:zh_TW
```

另外，如果某些时刻你必须上网点选某个网站以自动取得更新时，举例来说，早期的自动在线更新主机名称系统， 仅支持网页更新，那你如何进行更新呢？嘿嘿！ 可以使用 lynx 喔！ 利用 -dump 这个参数处理先：

```
[root@linux ~]# lynx -dump \
> http://some.site.name/web.php?name=user&password=pw > testfile
```

上面的网站后面有加个问号 (?) 对吧？后面接的则是利用网页的『 GET 』功能取得的各项变量数据， 利用这个功能，我们就可以直接点选到该网站上啰！非常的方便吧！而且会将执行的结果输出到 testfile 档案中，不过如果网站提供的数据是以『 POST 』为主的话，那鸟哥就不知道如何搞定了。 关于 GET 与 POST 的相关信息我们会在 WWW 服务器当中再次的提及的！别紧张啊！

---

wget

如果说 lynx 是在进行网页的『浏览』，那么 wget 就是在进行『网页数据的取得』。 举例来说，我们的 Linux 核心是放置在 www.kernel.ort 内，主要同时提供 ftp 与 http 来下载。 我们知道可以使用 lftp 来下载资料，但如果想要用浏览器来下载呢？ 那就利用 wget 吧！

```
[root@linux ~]# wget [option] [网址]
参数:
若想要联机的网站有提供账号与密码的保护时，可以利用这两个参数来输入喔！
--http-user=usrname
--http-password=password
--quiet : 不要显示 wget 在抓取数据时候的显示讯息
更多的参数请自行参考 man wget 吧！ ^_^

范例一：请下载 2.6.17 版的核心
[root@linux ~]# wget  \
> http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.17.tar.gz
--16:06:10--  http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.17.tar.gz
           => `linux-2.6.17.tar.gz'
Resolving www.kernel.org... 204.152.191.37, 204.152.191.5
Connecting to www.kernel.org|204.152.191.37|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 51,700,445 (49M) [application/x-gzip]

 3% [==>                         ] 1,890,568    220.69K/s    ETA 04:12
```

您瞧瞧～很可爱吧！不必透过浏览器，只要知道网址后，立即可以进行档案的下载， 又快速又方便，还可以透过 proxy 的帮助来下载呢！透过修改 /etc/wgetrc 来设定你的代理服务器：

```
[root@linux ~]# vi /etc/wgetrc
#http_proxy = http://proxy.yoyodyne.com:18023/  <==找到底下这几行，大约在 78 行后；
#ftp_proxy = http://proxy.yoyodyne.com:18023/
#use_proxy = on

# 将他改成类似底下的模样，记得，你必须要有可接受的 proxy 主机才行！
http_proxy = http://proxy.ncku.edu.tw:3128/
use_proxy = no
```

封包撷取功能

很多时候由于我们的网络联机出现问题，使用类似 ping 的软件功能却又无法找出问题点， 最常见的是因为路由与 IP 转递后所产生的一些困扰 (请参考防火墙与 NAT 主机部分)， 这个时候要怎么办？最简单的方法就是『分析封包的流向』啰！ 透过分析封包的流向，我们可以了解一条联机应该是如何进行双向的联机的动作， 也就会清楚的了解到可能发生的问题所在了！底下我们就来谈一谈这个 tcpdump 与图形接口的封包分析软件吧！

---

tcpdump

说实在的，对于 tcpdump 这个软件来说，你甚至可以说这个软件其实就是个黑客软件， 因为他不但可以分析封包的流向，连封包的内容也可以进行『监听』，如果你使用的传输数据是明码的话，不得了，在 router 上面就可能被人家监听走了！ 很可怕呐！所以，我们也要来了解一下这个软件啊！(注：这个 tcpdump 必须使用 root 的身份执行)

```
[root@linux ~]# tcpdump [-nn] [-i 接口] [-w 储存档名] [-c 次数] [-Ae]
                        [-qX] [-r 档案] [所欲撷取的数据内容]
参数：
-nn：直接以 IP 及 port number 显示，而非主机名与服务名称
-i ：后面接要『监听』的网络接口，例如 eth0, lo, ppp0 等等的界面；
-w ：如果你要将监听所得的封包数据储存下来，用这个参数就对了！后面接档名
-c ：监听的封包数，如果没有这个参数， tcpdump 会持续不断的监听，
     直到使用者输入 [ctrl]-c 为止。
-A ：封包的内容以 ASCII 显示，通常用来捉取 WWW 的网页封包资料。
-e ：使用资料连接层 (OSI 第二层) 的 MAC 封包数据来显示；
-q ：仅列出较为简短的封包信息，每一行的内容比较精简
-X ：可以列出十六进制 (hex) 以及 ASCII 的封包内容，对于监听封包内容很有用
-r ：从后面接的档案将封包数据读出来。那个『档案』是已经存在的档案，
     并且这个『档案』是由 -w 所制作出来的。
所欲撷取的数据内容：我们可以专门针对某些通讯协议或者是 IP 来源进行封包撷取，
     那就可以简化输出的结果，并取得最有用的信息。常见的表示方法有：
     'host foo', 'host 127.0.0.1' ：针对单部主机来进行封包撷取
     'net 192.168' ：针对某个网域来进行封包的撷取；
     'src host 127.0.0.1' 'dst net 192.168'：同时加上来源(src)或目标(dst)限制
     'tcp port 21'：还可以针对通讯协议侦测，如 tcp, udp, arp, ether 等
     还可以利用 and 与 or 来进行封包数据的整合显示呢！

范例一：以 IP 与 port number 捉下 eth0 这个网络卡上的封包，持续 3 秒
[root@linux ~]# tcpdump -i eth0 -nn
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
01:33:40.41 IP 192.168.1.100.22 > 192.168.1.11.1190: P 116:232(116) ack 1 win 9648
01:33:40.41 IP 192.168.1.100.22 > 192.168.1.11.1190: P 232:364(132) ack 1 win 9648
<==按下 [ctrl]-c 之后结束
6680 packets captured              <==捉下来的封包数量
14250 packets received by filter   <==由过滤所得的总封包数量
```

```
7512 packets dropped by kernel      <==被核心所丢弃的封包
```

如果你是第一次看 tcpdump 的 man page 时，肯定一个头两个大，因为 tcpdump 几乎都是分析封包的表头数据，使用者如果没有简易的网络封包基础，要看懂粉难呐！ 所以，至少您得要回到网络基础里面去将 TCP 封包的表头资料理解理解才好啊！ ^_^！至于那个范例一所产生的输出范例中，我们可以约略区分为数个字段， 我们以范例一当中那个特殊字体行来说明一下：

- 01:33:40.41：这个是此封包被撷取的时间，『时:分:秒』的单位；
- IP：透过的通讯协议是 IP ；
- 192.168.1.100.22 > ：传送端是 192.168.1.100 这个 IP，而传送的 port number 为 22，您必须要了解的是，那个大于 (>) 的符号指的是封包的传输方向喔！
- 192.168.1.11.1190：接收端的 IP 是 192.168.1.11， 且该主机开启 port 1190 来接收；
- P 116:232(116)：这个封包带有 PUSH 的数据传输标志， 且传输的数据为整体数据的 116~232 byte，所以这个封包带有 116 bytes 的数据量；
- ack 1 win 9648：ACK 与 Window size 的相关资料。

最简单的说法，就是该封包是由 192.168.1.100 传到 192.168.1.11，透过的 port 是由 22 到 1190 ，且带有 116 bytes 的数据量，使用的是 PUSH 的旗标，而不是 SYN 之类的主动联机标志。 呵呵！不容易看的懂吧！所以说，上头才讲请务必到 TCP 表头资料的部分去瞧一瞧的啊！

再来，一个网络状态很忙的主机上面，你想要取得某部主机对你联机的封包数据而已时， 使用 tcpdump 配合管线命令与正规表示法也可以，不过，毕竟不好捉取！ 我们可以透过 tcpdump 的表示法功能，就能够轻易的将所需要的数据独立的取出来。在上面的范例一当中，我们仅针对 eth0 做监听，所以整个 eth0 接口上面的数据都会被显示到屏幕上， 不好分析啊！那么我们可以简化吗？例如只取出 port 21 的联机封包，可以这样做：

```
[root@linux ~]# tcpdump -i eth0 -nn port 21
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
01:54:37.96 IP 192.168.1.11.1240 > 192.168.1.100.21: . ack 1 win 65535
01:54:37.96 IP 192.168.1.100.21 > 192.168.1.11.1240: P 1:21(20) ack 1 win 5840
01:54:38.12 IP 192.168.1.11.1240 > 192.168.1.100.21: . ack 21 win 65515
01:54:42.79 IP 192.168.1.11.1240 > 192.168.1.100.21: P 1:17(16) ack 21 win 65515
01:54:42.79 IP 192.168.1.100.21 > 192.168.1.11.1240: . ack 17 win 5840
01:54:42.79 IP 192.168.1.100.21 > 192.168.1.11.1240: P 21:55(34) ack 17 win 5840
```

瞧！这样就仅提出 port 21 的信息而已，且仔细看的话，你会发现封包的传递都是双向的， client 端发出『要求』而 server 端则予以『响应』，所以，当然是有去有回啊！ 而我们也就可以经过这个封包的流向来了解到封包运作的过程。 举例来说：

1. 我们先在一个终端机窗口输入『 tcpdump -i lo -nn 』 的监听，
2. 再另开一个终端机窗口来对本机 (127.0.0.1) 登入『ssh localhost』

那么输出的结果会是如何？

```
[root@linux ~]# tcpdump -i lo -nn
```

```
1 tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
2 listening on lo, link-type EN10MB (Ethernet), capture size 96 bytes
3 11:02:54.253777 IP 127.0.0.1.32936 > 127.0.0.1.22: S 933696132:933696132(0)
  win 32767 <mss 16396,sackOK,timestamp 236681316 0,nop,wscale 2>
4 11:02:54.253831 IP 127.0.0.1.22 > 127.0.0.1.32936: S 920046702:920046702(0)
  ack 933696133 win 32767 <mss 16396,sackOK,timestamp 236681316 236681316,nop,
  wscale 2>
5 11:02:54.253871 IP 127.0.0.1.32936 > 127.0.0.1.22: . ack 1 win 8192 <nop,
  nop,timestamp 236681316 236681316>
6 11:02:54.272124 IP 127.0.0.1.22 > 127.0.0.1.32936: P 1:23(22) ack 1 win 8192
  <nop,nop,timestamp 236681334 236681316>
7 11:02:54.272375 IP 127.0.0.1.32936 > 127.0.0.1.22: . ack 23 win 8192 <nop,
  nop,timestamp 236681334 236681334>
```

上表显示的头两行是 tcpdump 的基本说明，然后：

- 第 3 行显示的是『来自 client 端，带有 SYN 主动联机的封包』，
- 第 4 行显示的是『来自 server 端，除了响应 client 端之外(ACK)，还带有 SYN 主动联机的标志；
- 第 5 行则显示 client 端响应 server 确定联机建立 (ACK)
- 第 6 行以后则开始进入数据传输的步骤。

从第 3-5 行的流程来看，熟不熟悉啊？没错！那就是 三向交握 的基础流程啦！够有趣吧！ 不过 tcpdump 之所以被称为黑客软件之一可不止上头介绍的功能呐！ 上面介绍的功能可以用来作为我们主机的封包联机与传输的流程分析， 这将有助于我们了解到封包的运作，同时了解到主机的防火墙设定规则是否有需要修订的地方。

更神奇的使用要来啦！如果我们使用 tcpdump 在 router 上面监听『明码』的传输数据时， 例如 FTP 传输协议，你觉得会发生什么问题呢？ 我们先在主机端下达『 tcpdump -i lo port 21 -nn -X 』然后再以 ftp 登入本机，并输入账号与密码， 结果你就可以发现如下的状况：

```
[root@linux ~]# tcpdump -i lo -nn -X 'port 21'
    0x0000:  4500 0048 2a28 4000 4006 1286 7f00 0001  E..H*(@.@.......
    0x0010:  7f00 0001 0015 80ab 8355 2149 835c d825  .........U!I.\.%
    0x0020:  8018 2000 fe3c 0000 0101 080a 0e2e 0b67  .....<.........g
    0x0030:  0e2e 0b61 3232 3020 2876 7346 5450 6420  ...a220.(vsFTPd.
    0x0040:  322e 302e 3129 0d0a                      2.0.1)..

    0x0000:  4510 0041 d34b 4000 4006 6959 7f00 0001  E..A.K@.@.iY....
    0x0010:  7f00 0001 80ab 0015 835c d825 8355 215d  .........\.%.U!]
    0x0020:  8018 2000 fe35 0000 0101 080a 0e2e 1b37  .....5.........7
    0x0030:  0e2e 0b67 5553 4552 2064 6d74 7361 690d  ...gUSER.dmtsai.
    0x0040:  0a                                       .

    0x0000:  4510 004a d34f 4000 4006 694c 7f00 0001  E..J.O@.@.iL....
```

```
        0x0010:  7f00 0001 80ab 0015 835c d832 8355 217f  .........\.2.U!.
        0x0020:  8018 2000 fe3e 0000 0101 080a 0e2e 3227  .....>........2'
        0x0030:  0e2e 1b38 5041 5353 206d 7970 6173 7377  ...8PASS.mypassw
        0x0040:  6f72 6469 7379 6f75 0d0a                 ordisyou..
```

上面的输出结果已经被简化过了，你必须要自行在你的输出结果当中搜寻相关的字符串才行。 从上面输出结果的特殊字体中，我们可以发现『该 FTP 软件使用的是 vsftpd ，并且使用者输入 dmtsai 这个账号名称，且密码是 mypasswordisyou』 嘿嘿！你说可不可怕啊！如果使用的是明码的方式来传输你的网络数据？所以我们才常常在讲啊，网络是很不安全低！

另外你得了解，为了让网络接口可以让 tcpdump 监听，所以执行 tcpdump 时网络接口会启动在 『错乱模式 (promiscuous)』，所以你会在 /var/log/messages 里面看到很多的警告讯息， 通知你说你的网络卡被设定成为错乱模式！别担心，那是正常的。 至于更多的应用，请参考 man tcpdump 啰！

例题：如何使用 tcpdump 监听 (1)来自 eth0 适配卡且 (2)通讯协议为 port 22 ，(3)目标来源为 192.168.1.100 的封包资料？

答：

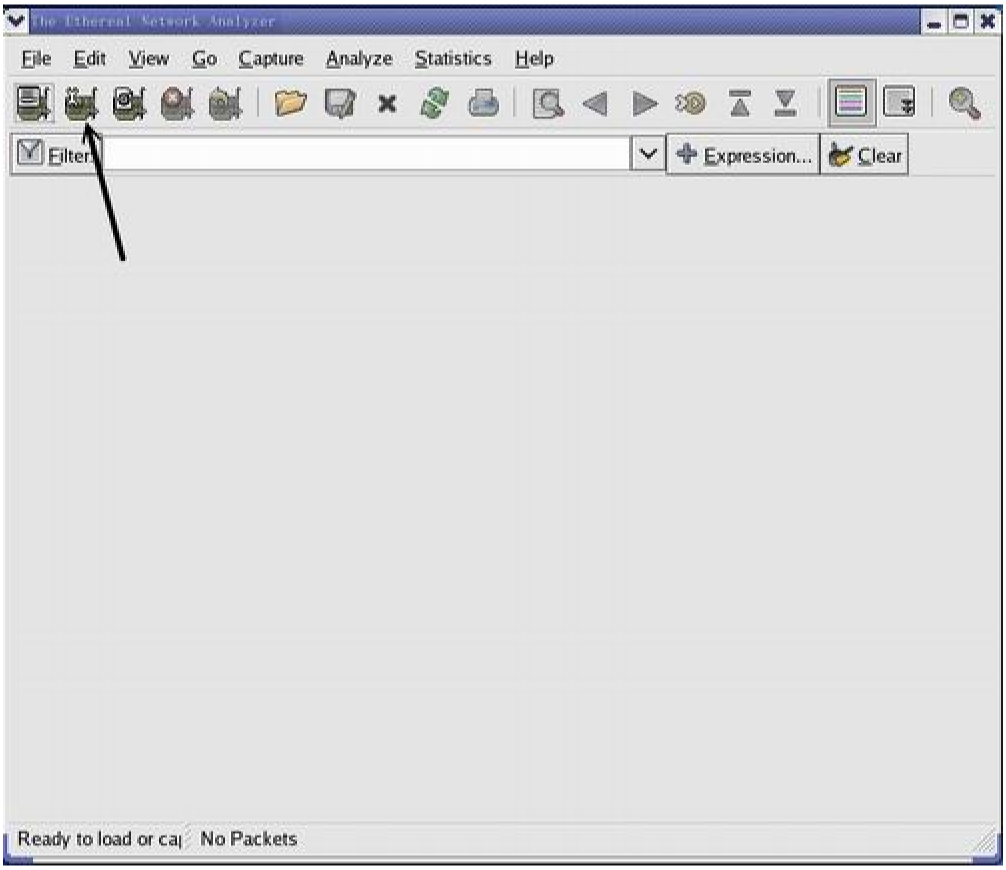
```
tcpdump -i eth0 -nn 'port 22 and src host 192.168.1.100'
```

---

ethereal

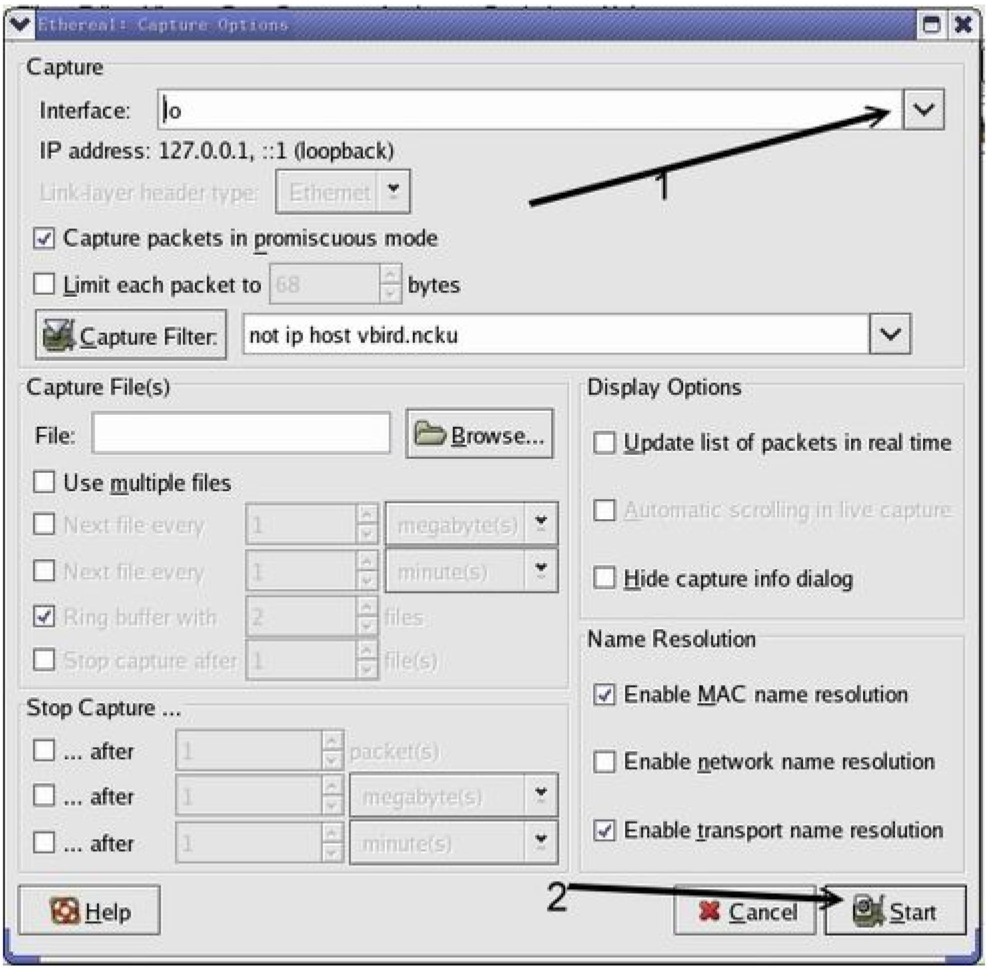
除了 tcpdump 这个软件之外，其实你还可以使用 ethereal 这个好用的网络流量分析软件呐！ ethereal 分为文字接口与图形接口，文字接口的用法与 tcpdump 相当的类似，不过他的指令名称为 tethereal 就是了。因为用法差不多，所以建议您直接使用 man tethereal 查阅吧！ 在 CentOS 上原本就有 ethereal 了，所以请拿出光盘来安装即可喔！ 需要安装 ethereal 与 ethereal-gnome 才行呐！

启动的方法很简单，你必须要在 X Window 底下，先开启一个终端机，然后直接输入 ethereal 后， 就会出现如下的画面了：



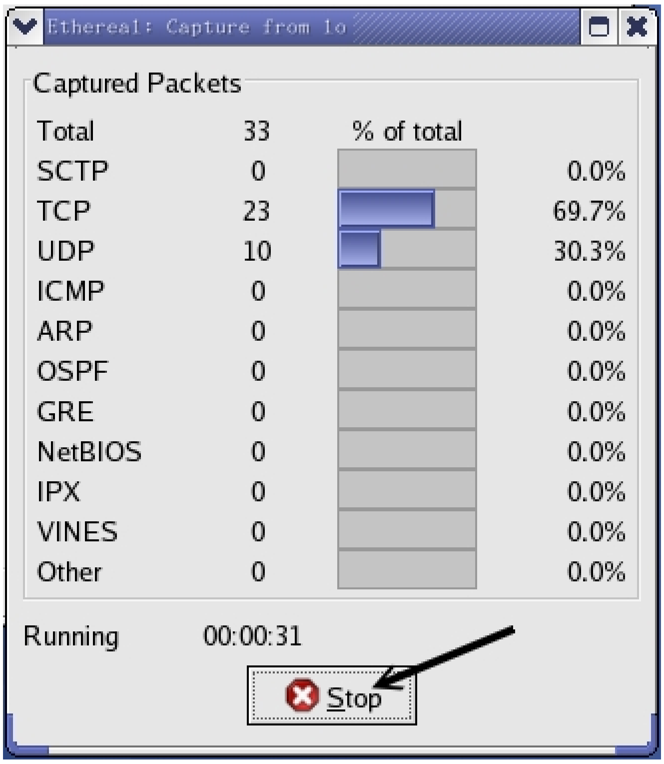
图五、ethereal 使用范例图

简单的作法，你可以点选如上图显示的那个按钮，会出现挑选监听的接口窗口，如下所示：



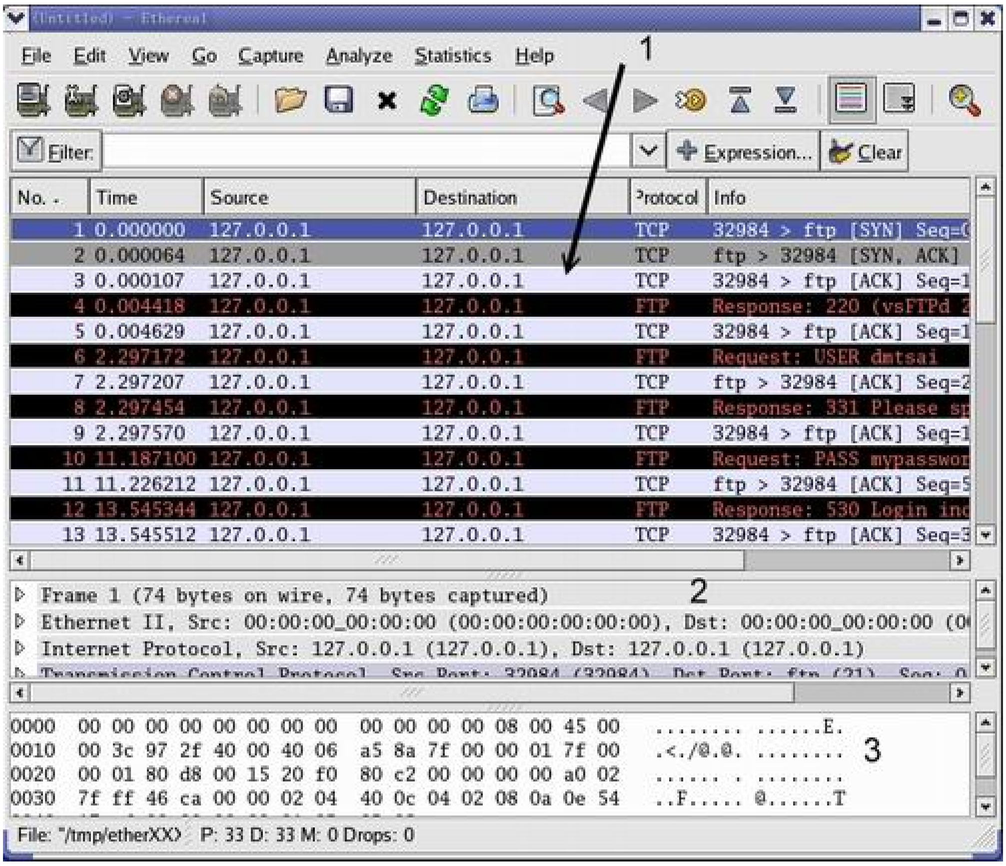
图六、ethereal 使用范例图

你应该选择要监听的接口，在这里因为是测试用的，所以鸟哥使用的是 lo 这个内部接口， 你当然应该要选择你自己的网络接口才是。然后按下 start 后，就会出现开始侦测的画面了：



图七、ethereal 使用范例图

在这个画面当中你可以看到很多类型的封包协议，在等你处理完毕后，就可以按下『stop』结束监听， 而开始进入如下的封包分析画面。



图八、ethereal 使用范例图

封包分析画面共分为三大区块，如上图所示，第一区块主要显示的是封包的标头资料， 内容就有点类似 tcpdump 的显示结果，第二区块则是详细的表头数据， 包括讯框的内容、通讯协议的内容以及 socket pair 等等信息。 第三区块则是 16 进位与 ASCII 码的显示结果。透过这个 ethereal 您就可以一口气得到所需要的所有封包内容啦！ 而且还是图形接口的，很方便吧！透过在第一区块选择不同的封包，就能够查阅每个封包的数据内容啰！

---

nc, netcat

这个 nc 可以用来作为某些服务的检测，因为他可以连接到某个 port 来进行沟通， 此外，还可以自行启动一个 port 来倾听其它用户的联机呐！非常的不错用！ 如果在编译的时候给予『GAPING_SECURITY_HOLE』参数的话，嘿嘿！ 这个软件还可以用来取得客户端的 bash 哩！可怕吧！我们的 CentOS 比较人性化，并没有给予上面的参数，所以我们不能够用来作为黑客软件～ 但是用来取代 telnet 也是个很棒的功能了！(有的系统将执行文件改名为 netcat 啦！)

```
[root@linux ~]# nc [IP|host] [port]
[root@linux ~]# nc -l -p [port]
参数：
-l ：作为监听之用，亦即开启一个 port 来监听用户的联机；
-p ：开启的这个 port number

范例一：连接本地端的 port 25 查阅相关讯息
```

```
[root@linux ~]# nc localhost 25
localhost.localdomain [127.0.0.1] 25 (smtp) open
220 pc.dm.tsai ESMTP Postfix
ehlo localhost
250-pc.dm.tsai
250-PIPELINING
250-SIZE 40000000
250-ETRN
quit
221 Bye
```

这个最简单的功能与 telnet 几乎一样吧！可以去检查某个服务啦！不过，更神奇的在后面， 我们可以建立两个联机来传讯喔！举个例子来说，我们先在 client 端的地方启动一个 port 来进行倾听：

```
范例二：激活一个 port 来监听使用者的联机要求
[root@linux ~]# nc -l -p 20000
# 启动一个 port 20000  在主机上，如果此时使用 netstat -tlnp
# 就可以看到系统上多出来一个 port 20000 在倾听使用者的联机喔！
```

然后在主机端的地方，也利用 nc 来联机到客户端，并且输入一些指令看看喔！

```
[root@linux ~]# nc localhost 20000
█  <==这里可以开始输入字符串了！
```

此时，在主机端我们可以打入一些字，你会发现在 client 端会同时出现你输入的字眼呐！ 如果你同时给予一些额外的参数，例如利用标准输入与输出 (stdout, stdin) 的话， 那么就可以透过这个联机来作很多事情了！ 当然 nc 的功能不只如此，你还可以发现很多的用途喔！ 请自行到您主机内的 /usr/share/doc/nc-1.10/scripts 目录下看看这些 script ，有帮助的呐！ 不过，如果你需要额外的编译出含有 GAPING_SECURITY_HOLE 功能， 以使两端联机可以进行额外指令的执行时，就得要自行下载原始码来编译了！

重点回顾

- 修改网络接口的硬件相关参数，可以使用 ifconfig 这个指令，包括 MTU 等等；
- ifup 与 ifdown 其实只是 script ，在使用时，会主动去 /etc/sysconfig/network-scripts 下找到相对应的装置设定文件，才能够正确的启动与关闭；
- 路由的修改与查阅可以使用 route 来查询，此外， route 亦可进行新增、删除路由的工作；
- ip 指令可以用来作为整个网络环境的设定，利用 ip link 可以修改『网络装置的硬件相关功能』，包括 MTU 与 MAC 等等，可以使用 ip address 修改 TCP/IP 方面的参数，包括 IP 以及网域参数等等， ip route 则可以修改路由！
- ping 主要是透过 ICMP 封包来进行网络环境的检测工作，并且可以使用 ping 来查询整体网域可接受最大的 MTU 值；
- 侦察每个节点的联机状况，可以使用 traceroute 这个指令来追踪！
- netstat 除了可以观察本机的启动接口外，还可以观察 Unix socket 的传统插槽接口数据；

- host 与 nslookup 预设都是透过 /etc/resolv.conf 内设定的 DNS 主机来进行主机名称与 IP 的查询;
- lftp 可以用来匿名登入远程的 FTP 主机;
- telnet 不只用来进行 BBS 的登入，也可以用来作为某些埠口的联机测试;
- lynx 主要的功能是『浏览』，包括本机上 HTML 语法的档案， wget 则主要在用来下载 WWW 的资料;
- 撷取封包以分析封包的流向，可使用 tcpdump ，至于图形接口的 ethereal 则可以进行更为详细的解析。
- 透过 tcpdump 分析三向交握，以及分析明码传输的数据，可发现网络加密的重要性。
- nc 可用来取代 telnet 进行某些服务埠口的检测工作,同时若自行编译 nc 时,可额外的执行 -e 参数。

课后练习

- 暂时将你的 eth0 这张网络卡的 IP 设定为 192.168.1.100 ，如何进行?

  ifconfig eth0 192.168.1.100

- 我要增加一个路由规则，以 eth0 连接 192.168.100.100/24 这个网域，应该如何下达指令?

  route add -net 192.168.100.0 netmask 255.255.255.0 dev eth0

- 我的网络停顿的很厉害，尤其是连接到 tw.yahoo.com 的时候，那么我应该如何检查那个环节出了问题?

  traceroute tw.yahoo.com

- 我发现我的 Linux 主机上面有个联机很怪异，想要将他断线，应该如何进行?

  以 root 的身份进行『netstat -anp |more』查出该联机的 PID，然后以『 kill -9 PID 』踢掉该联机。

- 您如何知道 green.ev.ncku.edu.tw 这部主机的 IP ?

  方法很多，可以利用 host green.ev.ncku.edu.tw 或 dig green.ev.ncku.edu.tw 或 nslookup green.ev.ncku.edu.tw 等方法找出

- 请找出您的机器上面最适当的 MTU 应该是多少?

  请利用『ping -c 3 -M do -s MTU yourIP 』找出您的 IP 的 MTU 数值。 事实上，你还可以先以 ip 设定网络卡较大的 MTU 后，在进行上述的动作，才能够找出网域内适合的 MTU。

- 如何在终端机接口上面进行 WWW 浏览?又该如何下载 WWW 上面提供的档案?

要浏览可以使用 lynx ，至于要下载则使用 wget 这个软件。

- 在终端机接口中，如何连接 bbs.sayya.org 这个 BBS ？

  利用 telnet bbs.sayya.org 即可连接上

- 请自行以 tcpdump 观察本机端的 ssh 联机时，三向交握的内容
- 请自行回答：为何使用明码传输的网络联机数据较为危险？并自行以软件将封包取出，并与同学讨论封包的信息
- 请自行至 Internet 下载 nc(netcat) 的原始码，并且编译成为具有 GAPING_SECURITY_HOLE 的参数， 然后建立一条联机使用 -e /bin/bash 尝试将本地端的 bash 丢给目的端执行 (特殊功能，可让 client 取得来自主机的 bash)。

参考数据

- 查询 MTU 的网站： http://forums.speedguide.net:8117/
- 在 Windows 底下修改 MTU 的方法：
  http://www.microsoft.com/taiwan/msclub/member/TIPS/Spring_2001/tip1to3/tip1to3_2.htm
- 本章各指令的 man page 说明文件

Linux 网络侦错

最近更新日期：2006/08/04

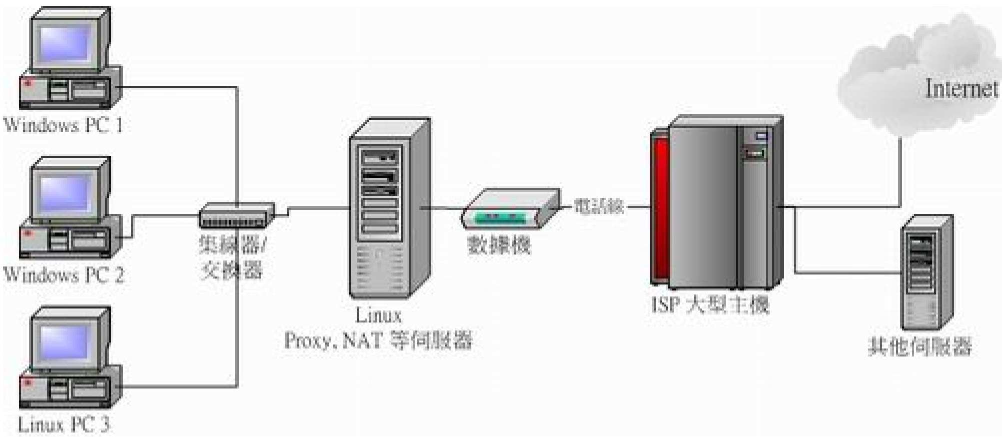
Linux 最强的，也是最让人称道的地方，就是他的网络功能了，不论是 Mail server、Web server、Proxy server 等等，都好好用喔！但是，我们也常在网络上看到一堆常见的问题，就是在问『我的 Linux 没有办法连上网络，该如何是好....』等等的问题，问来问去的重点大概都是一样的状况！伤脑筋！那鸟哥就把一些在 Linux 上面可能会发生的网络问题把他整理一下，看看您是不是有这方面的问题，参考看看吧！

无法联机原因分析

老是看到有朋友在网络上哀嚎说：『我的网络不通啊！』还有比较奇怪的是『啊怎么网络时通时不通』之类的问题， 这类的问题其实主要可以归类为硬件问题与软件设定问题，硬件的问题比较麻烦，因为需要透过一些专门的装置来分析硬件； 至于软件方面，绝大部分都是设定错误或者是观念错误而已，比较好处理啦！ OK！我们先来看看网络在哪里可能会出问题吧！

硬件问题

在网络基础章节当中我们曾提到很多的网络基础概念， 以及一些简单的硬件维护问题。以一个简单的星形联机来说，我们可以假设他的架构如同下图所示：



图一、局域网络的联机状态示意图

在上面的图示当中，"Linux PC3" 要连到 Internet 上面去的话，需要透过网络线、 交换器、 NAT 主机 (Linux 服务器或 IP 分享器)、ADSL 调制解调器，以及 Internet 上面的所有媒体设备 (包括路由器、桥接器、其它网络线等等)； 那么哪些地方可能会出问题啊？

1. 网络线材的问题：

   在上面的图标中，可以发现，其实网络接口设备中，使用最多的就是网络线啦！ 要注意网络线分成并行线与跳线 (RJ-45接头) ，而并不是所有的设备都支持自动分辨跳线与并行线的功能的！所以你必须要了解到你的设备 (Hub/Switch/调制解调器) 所支持的网络线；另外， 如果你的网络线是接在门缝处或者是容易凹折处，那很有可能具有被压毁的情况， 所以您需要注意一下这些事情：

   - 网络线被截断；
   - 网络线过度扭曲变形造成讯号不良；
   - 自制网络接头 (如 RJ-45 跳线头) 品质不良；
   - 网络接头与设备 (如 Hub)接触不良；

2. 网络卡、Hub 及 Router 等网络设备的问题：

   另外，还有一些网络设备也会有问题，常见的问题如下：

   - 网络卡不稳定、质量不佳，或者与整体系统的兼容度不佳；
   - 各网络设备的接头不佳，接触不良，造成讯号衰减；
   - 由于网络设备所在环境恶劣 (例如过热) 导致的当机问题；
   - 各网络设备使用方法不良，造成设备功能衰减；

3. 设备配置的规则：

   在各个设备的配置上是有一定的规则的，而最容易发生的问题就是太长的网络线会造成讯号的衰减， 导致网络联机的时间太长甚至无法联机。我们曾在网络基础当中谈过以太网络最长的支持距离， 还有一些其它网络媒体配置的问题您必须晓得的：

   - 使用错误的网络线，最常发生在并行线与跳线的分别！
   - 架设的网络线过长，导致讯号衰减太严重。例如以太网络 CAT5e 的线理论限制长度大概是在 90 公尺左右，若两个设备 (Hub/主机之间) 长度大于 90 公尺时，自然就容易出现讯号发生问题了！
   - 其它噪声的干扰，最常发生在网络线或者网络设备旁边有太强的磁波；
   - 局域网络上面，节点或者其它的设备太多，过去我们常以所谓的 543 原则来说明：
     - 5 个网段(segment)。所谓 segment 就在物理连接上最接近的一组计算机， 在一个 BNC 网段里面最多只能接 30 台计算机，且网线总长不能超过 185m。
     - 4 个增益器(repeater)。也就是将信号放大的装置。
     - 3 个计算机群体(population)。这个不好理解，也就是说前面所说的 5 个 segment 之中， 只能有 3 个可以装计算机，其它两个不行。

上述是一些最常见的硬件问题，当然啦，有的时候是设备本身就有问题， 而我们在网络基础里面谈到的那个很重要的『 网络布线 』的情况，也是造成网络停顿或通顺与否的重要原因呐！所以，硬件问题的判断比较困难点。 好～底下我们再来聊一聊软件设定的相关问题。

软件问题

所谓的软件问题，绝大部分就是 IP 设定错误啊，路由不对啊，还有 DNS 的 IP 设定错误等等的， 这些问题都是属于软件设定啦！只要将设定改一改，利用一些侦测软件查一查， 就知道问题出在哪里了！基本的问题有：

1. 网络卡的 IP 设定错误：
   例如：同一个 IP 在同一个网段中出现造成 IP 冲突、子屏蔽网络设定错误、网络卡的驱动程序使用错误、 网络卡的 IRQ、 I/O Address 的设定冲突等等；

2. 路由的问题 (route table)：
   最常见的就是预设路由 (default gateway) 设定错误了！或者是路由接口不符所导致的问题， 使得数据封包没有办法顺利的送出去。

3. 通讯协议不相符：
   最常发生在不同的操作系统之间的通讯传输，例如早期 Windows 98 与 Windows 2000 之间的『网芳』 若要达成沟通，则 Windows 98 必须要加装 NetBEUI 这个通讯协议才行。又例如两部 Linux 主机要透过 NFS 通讯协议传输数据时，两边都得要支持 portmap 这个启动 RPC 协议的程序才行！ 这些通讯协议我们都会在后面的章节分别介绍的啦！

4. 网络负荷的问题 (loading)：
   当同时有大量的数据封包涌进 Server 或者是 Hub 或者是同一个网域中， 就有可能造成网络的

停顿甚至挂点！另外，如果区网内有人使用 BT (P2P 软件) 或者是有人中毒导致蠕虫充满整个区网，也会造成网络的停顿问题；

5. 其它问题：
   例如：一些 port 被防火墙挡住了，造成无法执行某些网络资源；应用程序本身的 Bug 问题；应用程序中使用者的网络设定错误；以及不同的操作系统的兼容性问题等等。

问题的处理

既然问题发生了，就要去处理他啊！那如何处理呢？以上面的星形联机图示为例，把握两个原则：

- 先由自身的环境侦测起，可以由自身 PC 上的网络卡查起，到网络线、到 Hub 再到 ATU-R 等等的硬件先检查完。在这个步骤当中，最好用的软件就是 ping ， 而你最好能有两部以上的主机来进行联机的测试；
- 确定硬件没问题了，再来思考软件的设定问题！

实际上，如果网络不通时，你可以依序这样处理：

1. 了解问题：这个问题是刚刚发生？还是因为之前我做了什么动作而导致无法联机？
2. 确认 IP：先看看自己的网卡有无驱动？能否取得正确的 IP 来联机？
3. 确认区网联机：利用 ping 来沟通两部主机，确定网络线与中继的 hub/switch 工作正常；
4. 确认对外联机：看主机或 IP 分享器能否依据 连上 Internet 那一章的方法顺利取得 IP 参数，并以 ping 的方法确定对外联机是可以成功的；
5. 确认 DNS 查询：利用 nslookup 或 host 或 dig 检查 www.google.com 看看；
6. 确认 Internet 节点：可以利用 traceroute 检查各节点是否没问题？
7. 确认对方服务器正常服务：是否对方服务器忙线中？或他的机器挂了？
8. 确认我方服务器：是否某些服务没有正确启动？可利用 netstat 检查， 是否某些安全机制的套件没有开放，例如 SELinux 这项机制；
9. 防火墙或权限的问题：是否由于权限设定错误所致？ 是否由于您的机器有防火墙忘记启用可联机的埠口所致？这个可以透过 tcpdump 来处理！

透过这些处理动作后，一般来说，应该都可以解决您无法上网的问题了！ 当然啦，如果是硬件的问题，那么鸟哥也无法帮你，你可能最需要的是...... 『送修吧孩子！』

处理流程

既然知道上面已经谈到的几个小重点了，接下来当然是一个一个的给他处理掉啊！ 底下我们就得要一步一脚印的开始检查的流程啊！

步骤一：网络卡工作确认

其实，网络一出问题的时候，您应该从自己可以检查的地方检查起，因此， 最重要的地方就是检查您的网络卡是否有工作的问题啦！检查网络卡是否正常工作的方法如下：

1. 确定网络卡已经驱动成功：
   如果网络卡没有驱动成功，其它的，免谈！！所以你当然需要驱动你的网络卡才行！ 确认网络卡是否被驱动，可以利用 lspci 先看看有没有捉到 Ethernet 字样的显示信息，再以 dmesg 来检查是否被核心侦测到，最后使用 lsmod 看看有没有相对应的模块已被加载。 整个步骤可以参考『连上 Internet－Linux 网络卡』那一个小节， 这里鸟哥不再说明了！ ^_^！不过你要注意的是，如果上述的检测方式都无法发现你的网络卡模块， 那肯定就是核心与核心模块不支持你的网络卡啊！那该怎办？参考 『连上 Internet之网卡编译』就对了！

2. 确定可以手动直接建立 IP 参数：
   在顺利的加载网络卡的模块，并且『取得网络卡的代号』之后，我们可以利用 ifconfig 或 ip 来直接给予该网络卡一个网络地址试看看！ 看能否给予 IP 设定呢？例如：

```
[root@linux ~]# ifconfig eth0 192.168.1.100
```

   来直接建立该网络卡的 IP ，然后直接输入 ifconfig 看能否查阅到刚刚设定好的参数即可。 如果可以建立起该 IP ，就以 ping 来检测看看：

```
[root@linux ~]# ping 192.168.1.100
```

   如果有响应的话，那表示这个网卡的设定应该是没有问题了！ 再来则是开始检测一下局域网络内的各个连接硬件啦！

Tips:
事实上要再次的重申，如果您的主机捉不到您主机上的网络卡 (通常是内建的网络芯片)， 那么最好买一张便宜的螃蟹卡先来凑合着用，『先求有！再求完美』， 不要一开始就挑战自己的耐心啊！ 拜托拜托！



步骤二：局域网络内各项连接设备检测

在确认完了最重要的网络卡设定之后，并且确定网络卡是正常的之后， 再接着下来则是局域网络内的网络连接情况了！假设您是按照 图一所设定的星形联机局域网络架构，那么你必须要知道整个『网域』的概念！

1. 关于网域的概念：
   你得清楚的知道图一中各主机与服务器可以互相沟通是因为他们在『同一个网域里面』， 所以，你要知道所谓的 192.168.1.0/24 这种网域的表达方式所代表的意义， 且子屏蔽网络 (Netmask) 的意义也得了解。如果忘记了，请回去网络基础再翻一翻。

2. 关于 Gateway 与 DNS 的设定：
   Gateway 与 DNS (在 /etc/sysconfig/network-scripts/ifcfg-eth0, /etc/resolv.conf 的设定) 最容易被搞混～ 这两个并非是填写你的 Linux 主机的 IP 喔！应该是要填写 IP 分享器 (或

NAT 主机) 的 IP 在 Gateway 中， 填写 168.95.1.1 在 DNS 的 IP 设定当中！不能够搞错啊！如果还是不清楚？回去网络基础看看吧！

3. 关于 Windows 端的工作群组与计算机名称：
   假如您还需要资源共享，那么您就必须在 windows 系统中开放档案分享， 并且建议所有的计算机将『工作群组』设定相同，但『计算机名称』则不能相同！

假设你的区网内所有的主机 IP 都设定正确了，那么接下来你就可以使用 ping 来测试两部区网内主机的联机， 这个联机的动作可以让你测试两部主机间的各项设备，包括网络线、Hub/Switch 等等的咚咚！ 如果无法测试成功，那就请了解一下：

1. IP 参数是否设定正确：
   再次强调，先决定 IP 是对的！鸟哥在上课的时候常常发现同学无法连到我的主机上， 一经使用 ifconfig 才发现他们与我的 IP 不在同一个网段内，就是会有这样的情况发生啊！唉～

2. 联机的线材问题：
   包括我们前面提到的网络线本身折损、过度缠绕造成的讯号衰减问题等等， 另外，有些比较旧的 Hub/Switch 或者是 ATU-R (ADSL 调制解调器，俗称的小乌龟是也) 由于没有 Auto MDI/MDIX 的功能，所以无法自动的分辨跳线与否， 那么当你插错网络线的时候，也就无法接通啦！这样了解乎？另外，早期我们常常会说， 最简单判断每部主机是否顺利连接到 Hub/Switch 可以透过连接到 Switch 上的灯号来判断， 不过，由于有时候网络线本身讯号不良，虽然灯号还是会亮，不过就是无法连接到 Switch 的情况 (鸟哥自己就曾发生过啊！)，此时，跟朋友借一条 OK 的网络线来测试看看吧！

3. 网卡或 Hub/Switch 本身出问题：
   有一次鸟哥无法在外部连接到鸟哥的主机，怀疑是挂点了，结果冲到主机所在办公室察看，咦！ 主机是好好的嘛！那怎么会无法联机呢？原因是.......室内环境通风不良，加上 Switch 所在处温度过高， 加上那部旧的 switch 『刚好』风扇坏了，哈！就这样『 switch 当机 』 在重新启动 switch (拔掉再插上电源线) 后就正常了。所以啰，很多情况都是会发生的， 而局域网络内的环境也很容易影响到联机质量啊！

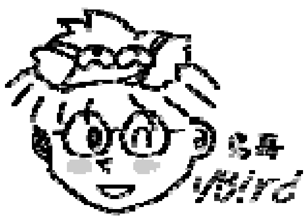
确定自己主机的 IP 与网卡没有问题，加上内部区网透过 ping 也测试过没有问题， 接下来就是要『取得可以对外联机的 IP 参数』啦！这个重要！

步骤三：取得正确的 IP 参数

什么叫『取得正确的 IP 参数』啊？还记得我们谈过如果要顺利的连接上 Internet 的话， 必须要可以跟 public IP 进行沟通才行，而与 public IP 取得沟通的方法，在台湾比较常见的有 ADSL, Cable modem, 学术网络, 电话拨接等等，在 CentOS 当中，我们可以透过修改 /etc/sysconfig/network-scripts/ifcfg-eth0，或者是利用 rp-pppoe 来进行拨接， 无论如何，你就是得要连接到某个 ISP 去就是了～在你确认所有的区网没有问题之后， 参考一下连上 Internet 那一章的介绍， 连上之后，立即以 ifconfig 看看有没有捉到正确的 IP 啊？ 在台湾如果使用 ADSL 联机的话，你应该可以顺利的取得一组正确的 Public IP 参数的！

Tips:

曾有国外的华人朋友来信说到，他们使用 ADSL 拨接之后竟然取得一组 Private IP ， 害他们没有办法架站！他们想请问这样的情况是否合理。如果您熟悉路由相关的概念之后， 当然会知道：『这当然合理！』，因为你取得的 IP 只是为了要连接到 ISP 去而已，而 ISP 与你的主机当然可以透过 Private IP 来联机啊！如果是这样的话，那么您就肯定无法架站了！ ^_^



另外，最常发现无法顺利取得 IP 的错误就是『BOOTPROTO』这个设定值设定错了！ 因为 static 与 hdcp 协议所产生的 IP 要求是不一样的啊！还记得吧！ 要特别留在 ifcfg-eth0 里面的设定参数喔！另外，如果你是使用 ADSL 拨接的， 但是老是无法拨接成功，那么建议你可以这样试看看：

- 将 ADSL 的调制解调器 (ATU-R) 整个关机，将 Switch/Hub 也关掉电源；
- 静待十分钟，等这些设备比较『凉快』一点后，再重新插上电源；
- 将 Linux 连接到 ADSL 的那块网卡 (假设为 eth0) 在 ifcfg-eth0 内，『ONBOOT』设定为 no，重新启动网络 (/etc/init.d/network restart)，然后再执行 adsl-start
- 如果还是无法拨接成功，并且你已经确认内部网域没有问题，那请中华电信的工程人员来帮忙您处理吧！

因为很多时候都是由于网络媒体过热，也有可能主机内部的一些网络参数有点问题， 所以，干脆就不要启动网卡，让 adsl-start 自动去启动网卡即可！ 如果顺利取得 IP 后，却还是无法顺利连到 Internet 上面时，你觉得还有哪些地方需要处理的呢？

步骤四：确认路由表的规则

如果你已经顺利取得正确的 IP 参数的话，那么接下来就是测试一下是否可以连上 Internet 啊！ 鸟哥建议你可以尝试使用 ping 来连连看 Hinet 的 DNS 主机，也就是 168.95.1.1 那部机器啦！

```
[root@linux ~]# ping -c 3 168.95.1.1
```

如果有响应，那就表示你的网络『基本上已经没有问题，可以连到 Internet 了！』， 那如果没有响应呢？明明取得了正确的 IP 却无法连接到外部的主机，肯定有鬼！呵呵！没错！ 还记得我们在网域内资料的传输可以直接透过 MAC 来传送，但如果不在区网内的数据，则需要透过路由，尤其是那个预设路由 (default route) 来帮忙转递封包吧！ 所以说，如果你的 public IP 无法连接到外部 (例如 168.95.1.1) ，可能的问题就出在路由与防火墙上面了。 假设你没有启动防火墙，那问题就缩小到剩下路由啰～

那路由的问题如何检查？就用 route -n 来检查啊！

例题：假设有个使用 ADSL 拨接的 Linux 主机，他的路由表如下，你觉得出了什么问题？

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
59.104.200.1	0.0.0.0	255.255.255.255	UH	0	0	0	ppp0
192.168.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
169.254.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	192.168.1.2	0.0.0.0	UG	0	0	0	eth0

答：

仔细看到上面的路由输出，第一条是 ppp0 产生的 public IP 接口，第二条是 eth0 的内部网域接口，在看到最后一条的 0.0.0.0/0.0.0.0 这个预设路由，竟然是内部网域的 eth0 为 gateway ？这不合理，最大的问题应该是出在 ifcfg-eth0 里面不小心设定了『GATEWAY=192.168.1.2』所致，解决的方法为：

1. 取消 ifcfg-eth0 内 GATEWAY=192.168.1.2 那一行，(该行亦可能出现在 /etc/sysconfig/network 内)
2. 重新启动网络 /etc/init.d/network restart
3. 重新进行拨接： adsl-stop; adsl-start

另外一个可能发生的情况，就是：『忘记设定预设路由』啦！ 例如使用 ifconfig 手动重新设定过网络卡的 IP 之后，其实路由规则是会被更新的， 所以预设路由可能就会不见了！那个时候你就得要利用 route add 来增加预设路由啰！

步骤五：主机名称与 IP 查询的 DNS 错误

如果你发现可以 ping 到 168.95.1.1 这个 Internet 上面的主机，却无法使用浏览器在网址列浏览 http://www.google.com 的话，那肯定 99% 以上问题是来自于 DNS 解析的困扰！ 解决的方法就是直接到 /etc/resolv.conf 去看看设定值对不对啊！ 一般常见的内容是这样的：

```
[root@linux ~]# vi /etc/resolv.conf
nameserver 168.95.1.1
nameserver 139.175.10.20
```

最常见的错误是『那个 nameserver 的拼字写错了！』真是最常见的问题～～另外，如果 client 端是 Windows 系统呢？常常初学者会搞错的地方就是在 windows 的设定了！要注意： Windows 端的 DNS 设定与主机端 /etc/resolv.conf 的内容相同即可！ 很多初学者都以为 TCP/IP 内的 DNS 主机是填上自己的 Linux 主机，这是不对的 (除非您自己的 Linux 上面有 DNS 服务) !您只要填上您的 ISP 给您的 DNS 主机 IP 位置就可以了

步骤六：Linux 的 NAT 服务器或 IP 分享器出问题

NAT 服务器最简单的功能就是 IP 分享器啦！NAT 主机一定是部路由器，所以你必须要在 Linux 上面观察好正确的路由信息。否则肯定有问题。另外， NAT 主机上面的防火墙设定是否合理？ IP 分享器上面是否有设定抵挡的机制等等，都会影响到对外联机是否能够成功的问题点。 关于 NAT 与防火墙我们会在后续的章节继续介绍的啦！

步骤七：Internet 的问题

Internet 也会出问题喔！当然啦～没有任何东西是不会出问题的！ 举例来说，好几年前台湾西岸因为施工的关系，导致南北网络骨干缆线被挖断， 结果导致整个 Internet 流量的大塞车！这就是 Internet 的

问题～还有，数年前 Study Area 网站放置的地点由于路由器设定出了点差错， 结果导致联机速度的缓慢。这都不是主机本身出问题，而是 Internet 上面某个节点出了状况。 想要确认是否问题来自 Internet 的话，就使用 traceroute 吧！ 查察看问题是来自那个地方再说！

步骤八：主机的问题

如果上述的处理都 OK ，却无法登入某部主机时，我想，最大的问题就是出现在主机的设定啦！ 这包括有：

- 主机并没有开放该项服务：例如主机关闭了 telnet ，那你使用 telnet 去联机，是无法连接上的啦！
- 主机的权限设定错误：例如你将某个目录设定为 drwx------ ，该目录拥有者为 root ， 你却将该目录开放给 WWW 来浏览，由于 WWW 无法进入该目录，所以当然无法正确的给客户端浏览啊！这是最典型的权限设定错误的情况啊！
- 安全机制设定错误：例如 SELinux 是用来更细微控管主机存取的一种核心机制， 如果你没有设定好就启用的话，那么主机的服务很多都『无法顺利的启用』， 关闭 SELinux 就好了。而其它例如 /etc/hosts.deny, PAM 模块等等， 都可能造成使用者无法登入的问题！这就不是网络问题，而是主机造成联机无法成功！
- 防火墙问题：防火墙设定错误也是一个很常见的问题，你可以使用 tcpdump 来追踪封包的流向，以顺利的了解防火墙是否设定错误。

基本上，一个网络环境的检测工作可不是三言两语就讲的完的～而且常常牵涉到很多经验的问题～ 请您常常到一些讲座的场合去听听看大家的经验，去 google 看看人家的解决方法， 都有助于让你更轻易的解决网络问题的喔！ ^_^

课后练习

- 以图一的星形联机为例，你的 Linux PC 3 可以 ping 到 Windows PC1 ，但是反过来， Windows PC1 无法 ping 到 Linux PC3，你觉得原因可能发生在哪里？

  由于两边已经可以用 ping 进行联机，所以硬件应该是没有问题了。 而 Linux --> Windows 没问题， Windows --> Linux 有问题，可能是由于 Linux 主机上面的防火墙所致。 可以使用 iptables -L -n 去查阅一下防火墙的设定规则。详细的防火墙请参考后续的章节。

主机基本安全之一：限制 Linux 对外联机的埠口

最近更新日期：2006/08/08

为什么我们的主机会响应网络上面的一些要求封包呢？例如我们设定了一部 WWW 主机后，当有来自 Internet 的 WWW 要求时，我们的主机就会予以响应，这是因为我们的主机有启用了 WWW 的监听埠口 (port) 啊！这里就要特别留意了，当我们启用了一个 daemon 时，就可能会造成主机的 Port 在进行 Listen 的动作，此时该 daemon 就是已经对网络上面提供服务了！万一这个 daemon 有漏洞，因为他提供 Internet 的服务，所以就容易被 Internet 上面的 cracker 所入侵了！所以说，仔细的检查自己系统上面的 port 到底开了多少个，并且予以严格的管理， 才能够降低被入侵的可能性啊！

Linux 的埠口 (port)

我们在网络基础的通讯协议那个小节曾经谈到 TCP 封包表头最重要的就是来源与目标的端口口 (port) 了，若再加上来源与目标的 IP 就可成为一组 Socket pair ，这个 port 就是用在网络联机时提供联机接口的咚咚啰，在开始这一节之前，请您先前往网络基础那一章再瞧一瞧先。 除了这个之外，还有没有其它需要注意的事项呢？底下我们就来谈一谈先！

什么是 port

你或许常常会在网络上听说『我的主机开了多少的 port ，会不会被入侵呀？』 或者是说『开那个 port 会比较安全？又，我的服务应该对应什么 port 呀？』呵呵！很神奇吧！怎么一部主机上面有这么多的奇怪的 port 呢？

其实也不怎么难啦！在网络基础里面我们曾经介绍过很多的网络概念， 所以你会知道要达成一条 server/client 的联机，需要一组 Socket pair 来建立联机， 这也就是说，网络联机是『双向』的。 此外，既然我们想要联机到主机端，那么主机势必得要启动一个大家都知道的 port 在『监听』吧， 否则如何达成联机呢？您说是吧！另外， client 端是否要启用固定的 port 来联机？ 当然不需要啊～那共有多少 port 呢？底下我们就先来谈一谈。

- 主机端的监听 (Listen)：
  想一想，你要如何连上 Yahoo 的网站去看新闻？首先当然是要打开浏览器，然后输入 Yahoo 的网址，之后我们的浏览器就会连接到 Yahoo 的 WWW 网站去要求数据了。既然如此的话，那么那部 Yahoo 的 WWW 主机当然就得要启动 WWW 的服务啦， 然后我们的浏览器才能够连接到该服务。

这也就是说『主机所启用的 port 其实是由某些网络服务 (program) 所启动的』。而为了连接上的方便，因此很多服务所开启的 port 是固定的，例如 WWW 开启在 port 80 ，mail 开启在 port 25 等等；

- 客户端的 port：
  客户端启动的 port 是随机产生的， 主要是开启在大于 1024 以上的埠口，这个 port 也是由某些软件所产生的，例如上面提到的例子， 我们的浏览器想要连接到 Yahoo 的 WWW 主机，那么浏览器就得要启用一个 port 来与主机进行联机， 以组成一组 Socket pair 来传输数据嘛！

所谓的『监听』是某个服务程序会一直常驻在内存当中，所以该程序启动的 port 就会一直存在。至于 port 在传输过程中的判断，那就由 TCP/UDP 等通讯协议的表头数据来记录的啊，我们的主机透过分析 TCP/UDP 的表头数据就能够了解到该联机所需要连接的软件是那个， 而给予正确的数据响应。所以，一部主机上面当然可以同时启动很多不同的服务啊！ ^_^。

还有上面提到的一些重点你也得再了解一下，那就是：

- 共 65536 个 port：
  预设的情况下，我们的主机会有 65536 个 port，而这些 port 又分成两个部分，以 port 1024 作区隔：

- 只有 root 才能启动的保留的 port：
  在小于 1023 (连同 1023) 的埠口，都是需要以 root 的身份才能启动的， 这些 port 主要是用于一些常见的通讯服务，在 Linux 系统下，常见的协议与 port 的对应是记录在 /etc/services 里面的。 一般来说，这些 port 最好保留给一些预设的服务来使用，不要自己随意使用到这些 port， 因为这些 port 是目前 Internet 上面所惯用的，所以一些程序开发者在进行软件的开发时， 就能够针对这些 well know 的埠口直接来开发，大家也比较容易使用服务器的功能啊！

- 大于 1024 用于 client 端的 port：
  在大于 1024 以上的 port 主要是作为 client 端的软件启动的 port 。这些 port 几乎都是依序随机使用的， 例如前面谈到的浏览器，就是使用大于 1024 以上的 port。那如果用到 port 65535 后， 系统会主动再由前面没有使用到的埠口 (如 1024) 再重新依序使用。

- 是否需要三向交握：
  建立可靠的联机服务需要使用到 TCP 协议，也就需要所谓的 三向交握了， 如果是非可靠的联机服务，例如 DNS ，那只要使用 UDP 协议即可。

- 通讯协议启用在非正规的 port：
  我们知道浏览器预设会连接到 WWW 主机的 port 80，那么你的 WWW 是否可以启动在非 80 的其它埠口？ 当然可以啊！你可以透过 WWW 软件的设定功能将该软件使用的 port 启动在非正规的埠口， 只是如此一来，您的客户端要连接到你的主机时，就得要在浏览器的地方额外指定你所启用的非正规的埠口才行。 这个启动在非正规的端口口功能，常常被用在一些所谓的地下网站啦！ ^_^。另外， 某些软件预设就启动在大于 1024 以上的端口口，如 MySQL 数据库软件就启动在 3306。

- 所谓的 port 的安全性：
  事实上，没有所谓的 port 的安全性！因为『Port 的启用是由服务软件所造成的』， 也就是说，

真正影响网络安全的并不是 port ，而是启动 port 的那个软件 (程序)！ 或许你偶而会听到：『没有修补过漏洞的 bind 8.x 版，很容易被黑客所入侵，请尽快升级到 bind 9.x 以后版本』，所以啰，对安全真正有危害的是『某些不安全的服务』 而不是『开了哪些 port 』才是！因此，没有必要的服务就将他关闭吧！ 尤其某些网络服务还会启动一些 port 哩！另外，那些已启动的软件也需要持续的保持更新喔！

观察 port

好了，我们现在知道这个 port 是什么鬼东西了，再来就是要去『看他到底在干啥？』没错！ 再来就是要来了解一下，我们的主机到底是开了多少的 port 呢？如同我们前面说的，您得要先了解一下，我们的『服务』跟『 port 』对应的档案是哪一个？再提醒一次呦！是『 /etc/services 』啦！ 而常用来观察 port 的则有底下两个程序：

- netstat：在本机上面以自己的程序监测自己的 port；
- nmap：透过网络的侦测软件辅助，可侦测非本机上的其它网络主机，但有违法之虞。

见他的大头王！怎么使用 nmap 会违法？呵呵！由于 nmap 的功能太强大了，所以很多 cracker (怪客，网络上面的闲人) 会直接以他来侦测别人的主机，这个时候就可能造成违法啦！ 只要您使用 nmap 的时候不要去侦测别人的计算机主机，那么就不会有问题啦！ 底下我们分别来说一说这两个宝贝吧！

- netstat

在做为主机的 Linux 系统中，开启的网络服务越少越好！ 因为较少的服务可以较容易除错 (debug) 与了解安全漏洞，并可避免不必要的入侵管道！ 所以，这个时候请了解一下您的系统当中有没有哪些服务被开启了呢？ 要了解自己的系统当中的服务项目，最简便的方法就是使用 netstat 了!这个东西不但简单 (每一部 Linux 机器当中预设都会安装的程序喔!)，而且功能也是很不错的。 这个指令的使用方法在 Linux 常用网络功能指令介绍当中提过了， 底下我们仅提供如何使用这个工具的方法啰！

列出在监听的网络服务：
列出网络服务的方式简单，如下所示：

```
[root@linux ~]# netstat -tunl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address       Foreign Address     State
tcp        0      0 0.0.0.0:25          0.0.0.0:*           LISTEN
tcp        0      0 :::80               :::*                LISTEN
tcp        0      0 :::22               :::*                LISTEN
tcp        0      0 :::25               :::*                LISTEN
```

上面说明了我的主机有启动 port 25, 80, 22 等，而且观察各联机接口，可发现这三个 port 都有对外提供联机的能力喔！

列出已联机的网络联机状态：
如果仅是要列出网络接口上已经联机的或者是一些联机过程挂断、连接程序的网络状态， 可以使用如下的方式来处理：

```
[root@linux ~]# netstat -tun
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address         Foreign Address      State
tcp        0      0 192.168.10.100:25     192.168.10.5:3151    TIME_WAIT
tcp        0      0 192.168.10.100:22     192.168.10.150:1832  ESTABLISHED
```

从上面的数据来看，我的主机 (192.168.10.100) 目前仅有一条已建立的联机， 那就是与 192.168.10.150 那部主机连接的联机，并且联机方线是由对方连接到我主机的 port 22 来取用我主机的服务呐！至于那个 TIME_WAIT 则是在等待该联机挂断啦！

删除已建立或在监听当中的联机：

如果想要将已经建立，或者是正在监听当中的网络服务关闭的话，最简单的方法当然就是找出该联机的 PID， 然后将他 kill 掉即可啊！例如下面的范例：

```
[root@linux ~]# netstat -tunp
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address      Foreign Address      State       PID/P name
tcp        0     68 192.168.10.100:22 192.168.10.150:1832 ESTABLISHED 13247/sshd
```

如上面的范例，我们可以找出来该联机是由 sshd 这个程序来启用的，并且他的 PID 是 13247， 希望你不要心急的用 killall 这个指令，否则容易删错人 (因为你的主机里面可能会有多个 sshd 存在)， 应该要使用 kill 这个指令才对喔！

```
[root@linux ~]# kill -9 13247
```

- nmap

如果你要侦测的设备并没有自己的操作系统，举例来说，你想要了解一下公司的网络打印机是否有开放某些协议时， 那该如何处理啊？现在你知道 netstat 可以用来查阅本机上面的许多监听中的通讯协议， 那例如网络打印机这样的非本机的设备，要如何查询啊？呵呵！用 nmap 就对了！

nmap 的套件说明之名称为：『Network exploration tool and security scanner』，顾名思义， 这个东西是被系统管理员用来管理系统安全性查核的工具！他的具体描述当中也提到了， nmap 可以经由程序内部自行定义的几个 port 对应的指纹数据，来查出该 port 的服务为何，所以我们也可以藉此了解我们主机的 port 到底是干嘛用的！如果您是安装 Linux 是 Red Hat 系统的话，那么这个 nmap 套件应该已经安装妥当了，万一没有这个套件的话，也可以来到底下的网站下载：

- http://insecure.org/nmap/

```
[root@linux ~]# nmap [扫瞄类型] [扫瞄参数] [hosts 地址与范围]
参数：
[扫瞄类型]：主要的扫瞄类型有底下几种：
    -sT：扫瞄 TCP 封包已建立的联机 connect() ！
    -sS：扫瞄 TCP 封包带有 SYN 卷标的数据
    -sP：以 ping 的方式进行扫瞄
    -sU：以 UDP 的封包格式进行扫瞄
```

```
    -sO：以 IP 的协议 ( protocol ) 进行主机的扫瞄
[扫瞄参数]：主要的扫瞄参数有几种：
    -PT：使用 TCP 里头的 ping 的方式来进行扫瞄，可以获知目前有几部计算机存活(较常用)
    -PI：使用实际的 ping (带有 ICMP 封包的) 来进行扫瞄
    -p ：这个是 port range ，例如 1024-, 80-1023, 30000-60000 等等的使用方式
[Hosts 地址与范围]：这个有趣多了，有几种类似的类型
    192.168.0.100  ：直接写入 HOST IP 而已，仅检查一部；
    192.168.0.0/24 ：为 C Class 的型态，
    192.168.*.*    ：嘿嘿！则变为 B Class 的型态了！扫瞄的范围变广了！
    192.168.0.0-50,60-100,103,200 ：这种是变形的主机范围啦！很好用吧！

范例一：使用预设参数扫瞄本机所启用的 port
[root@linux ~]# nmap localhost
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
80/tcp   open  http
139/tcp  open  netbios-ssn
# 在预设的情况下， nmap 仅会扫瞄 TCP 的协议喔！
```

nmap 的用法很简单呐！就直接在指令后面接上 IP 或者是主机名称即可。不过，在预设的情况下 nmap 仅会帮你分析 TCP 这个通讯协议而已，像上面这个例子，他只会帮我列出 4 个已经开启的 TCP 的端口口号码， 但优点是顺道也将开启该埠口的服务也列出来了，真是好！ ^_^！那如果想要同时分析 TCP/UDP 这两个常见的通讯协议呢？可以这样做：

```
[root@linux ~]# nmap -sTU localhost
PORT     STATE         SERVICE
22/tcp   open          ssh
25/tcp   open          smtp
80/tcp   open          http
137/udp  open|filtered netbios-ns
138/udp  open|filtered netbios-dgm
139/tcp  open          netbios-ssn
```

嘿嘿！与前面的范例比较一下，你会发现这次多了两个 UDP 的埠口，分别是 137 与 138 ， 这样分析好多了吧！然后，如果你想要了解一下到底有几部主机活在你的网络当中时， 则可以这样做：

```
[root@linux ~]# nmap -sP 192.168.10.0/24
Host 192.168.10.171 appears to be up.
MAC Address: 00:01:E6:B3:AA:CC (Hewlett-Packard Company)
Host 192.168.10.174 appears to be up.
MAC Address: 00:04:75:FF:CC:DD (3 Com)
Host 192.168.10.175 appears to be up.
MAC Address: 00:0C:6E:BA:11:22 (Asustek Computer)
```

看到否？你的环境当中有三部主机活着呐！并且该 IP 所对应的 MAC 也会被记录下来， 很不错吧！如果你还想要将各个主机的启动的 port 作一翻侦测的话，那就得要使用：

```
[root@linux ~]# nmap 192.168.10.0/24
```

之后你就会看到一堆 port number 被输出到屏幕上啰～如果想要随时记录整个网段的主机是否不小心开放了某些服务， 嘿嘿！利用 nmap 配合数据流重导向 (>, >> 等) 来输出成为档案， 那随时可以掌握住您局域网络内每部主机的服务启动状况啊！ ^_^

请特别留意，这个 nmap 的功能相当的强大，也是因为如此， 所以很多刚在练习的黑客会使用这个软件来侦测别人的计算机，这个时候请您特别留意， 目前很多的人已经都有『特别的方式』来进行登录的工作！例如以 TCP_Wrappers (/etc/hosts.allow, /etc/hosts.deny) 的功能来记录曾经侦测过该 port 的 IP！这个软件用来『侦测自己机器的安全性』是很不错的一个工具，但是如果用来侦测别人的主机， 可是会『吃上官司』的！特别留意！！

Port 的启动与关闭

现在你知道其实 port 是由某些程序所启动的，所以要关闭某些 port 时，那就直接将某个程序给他关闭就是了！ 那关闭的方法你当然可以使用 kill ，不过，毕竟不是正统的解决之道，因为 kill 这个指令通常具有强制关闭某些程序的功能，但我们想要正常的关闭该程序啊！所以，就利用系统给我们的 script 来关闭就好了啊。 在此同时，我们就得再来稍微复习一下，一般传统的服务有哪几种类型？

---

stand alone 与 super daemon

我们在鸟哥的 Linux 私房菜 -- 基础学习篇内谈到， 在一般正常的 Linux 系统环境下，服务的启动与管理主要有两种方式：

- stand alone
  顾名思义，stand alone 就是直接执行该服务的执行档，让该执行文件直接加载到内存当中运作，用这种方式来启动可以让该服务具有较快速响应的优点。一般来说，这种服务的启动 script 都会放置到 /etc/init.d/ 这个目录底下，所以你通常可以使用：『 /etc/init.d/sshd restart 』之类的方式来启动这种服务；

- Super daemon
  用一个超级服务作为总管，以管理一些网络服务。在 CentOS 4.3 里面使用的则是 xinetd 这个 super daemon 啊！这种方式启动的网络服务虽然在响应上速度会比较慢， 不过，可以透过 super daemon 额外提供一些控管，例如控制何时启动、何时可以进行联机、 那个 IP 可以连进来、是否允许同时联机等等。通常设定档放置在 /etc/xinetd.d/ 当中，但设定完毕后需要重新以『 /etc/init.d/xinetd restart 』重新来启动才行！

关于更详细的服务说明，请参考基础篇的 认识服务 一文， 鸟哥在这里不再赘述。好，那么如果我想要将我系统上面的 port 25 关掉的话， 那应该如何关闭呢？最简单的作法就是先找出那个 port 25 的启动程序喔！

```
[root@linux ~]# netstat -tnlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State    PID/Program name
```

```
tcp        0      0 127.0.0.1:25   0.0.0.0:*       LISTEN  2030/master
tcp        0      0 :::22          :::*            LISTEN  1961/sshd
# 咦！怎么会是 master 这个玩意儿？用 which 这个指令还找不到这个 master
# 那怎办？没关系，我们可以透过 locate 配合正规表示法找到这个指令的！

[root@linux ~]# locate master | grep '/master$'
/usr/libexec/postfix/master
# 嘿嘿！那个正规表示法就可以找到上述的输出结果，然后再由 rpm 来处理！

[root@linux ~]# rpm -qf /usr/libexec/postfix/master
postfix-2.2.2-2
# 找到了！就是这个套件！所以将他关闭的方法可能就是：

[root@linux ~]# rpm -qc postfix | grep init
/etc/rc.d/init.d/postfix
[root@linux ~]# /etc/init.d/postfix stop
```

透过上面的这个分析的流程，你可以利用系统提供的很多方便的工具来达成某个服务的关闭！ 为啥这么麻烦？不是利用 kill -9 2030 就可以删掉该服务了吗？ 是没错啦！不过，你知道该服务是做啥用的吗？你知道将他关闭之后，你的系统会出什么问题吗？ 如果不知道的话，那么利用上面的流程不就可以找出该服务套件，再利用 rpm 查询功能， 不就能够知道该服务的作用了？所以说，这个方式还是对您会有帮助的啦！ 底下请您试着将您 CentOS 或者是其它版本的 Linux 的 Telnet 打开试看看。

例题：我们知道系统的 Telnet 服务通常是以 super daemon 来控管的，请您启动您系统的 telnet 试看看。

答：

1. 要启动 telnet 首先必须要已经安装了 telnet 的服务器才行，所以请先以 rpm 查询看看是否有安装 telnet-server 呢？『rpm -qa | grep telnet-server』如果没有安装的话，请利用原版光盘来安装，或者使用『yum install telnet-server』安装一下先；
2. 由于是 super daemon 控管，所以请编辑 /etc/xinetd.d/telnet 这个档案，将其中的『disable = yes』改成『disable = no』之后以『/etc/init.d/xinetd restart』重新启动 super daemon 吧！
3. 利用 netstat -tnlp 察看是否有启动 port 23 呢？

设定开机时启动服务

如果刚刚你已经利用类似前一节的方法将一些服务关闭了，但是下次再重新开机后，咦！ 怎么那些被关闭的服务又『春风吹又生』的给他『长』出来了？呵呵～没错啊， 因为前一节的作法是可以立即将某个服务关闭，但是却不会影响到开机时是否会启动与否的设定。 唉～伤脑筋～

如果你想要在开机的时候就启动或不启动某项服务时，那就得要了解一下 基础学习篇里面谈到的开机流程管理 的内容啦！ 在 Unix like 的系统当中我们都是透过 run level 来设定某些执行等级需要启动的服务， 以 Red Hat 系统来说，这些 run level 启动的数据都是放置在 /etc/rc.d/rc[0-6].d/ 里面的，那如何管理该目录下的 script 呢？ 手动处理吗？会疯掉的呐！所以你必须要熟悉 chkconfig 或 Red Hat 系统的 ntsysv 这几个指令才行！

Tips:
这几个指令不熟吗？这个时候鸟哥不得不说了：『有 man 堪用直需用，莫待无 man 空自猜』赶紧给他 man 下去啦！



例题：(1)如何查阅 portmap 这个程序一开机就执行？ (2)如果开机就执行，如何将他改为开机时不要启动？ (3)如何立即关闭这个 portmap 服务？

答：

1. 可以透过『 chkconfig --list | grep portmap 』与『 runlevel 』确认一下你的环境与 portmap 是否启动？
2. 如果有启动，可透过『 chkconfig --level 35 portmap off 』来设定开机时不要启动；
3. 可以透过『 /etc/init.d/portmap stop 』来立即关闭他！

聪明的你一定会问说：『鸟哥，你的意思是只要将系统所有的服务都关闭，那系统就会安全啰？』 当然.... 不是！因为『很多的系统服务是必须要存在的，否则系统将会出问题』 举例来说，那个保持系统可以具有工作排程的 crond 服务就一定要存在，而那个记录系统状况的 syslogd 也当然要存在～否则怎知道系统出了啥问题？ 底下鸟哥列出几个常见的必须要存在的系统服务给大家参考参考先！这些服务请不要关闭啊！

服务名称	服务内容
acpid	新版的电源管理模块，通常建议开启，不过，某些笔记型计算机可能不支持此项服务，那就得关闭
atd	在管理单一预约命令执行的服务，应该要启动的
crond	在管理工作排程的重要服务，请务必要启动啊！
iptables	Linux 内建的防火墙软件，这个也可以启动啦！
keytables	如果你的键盘非正规的格式时，这个服务的启动或许可以帮助你喔！
network	这个重要了吧？要网络就要有他啊！
sshd	这是系统预设会启动的，可以让你在远程以文字型态的终端机登入喔！
syslog	系统的登录文件记录，很重要的，务必启动啊！

xinetd	就是那个 super daemon 嘛！所以也要启动啦！
xfs	用来管理 X Window 字形数据的服务，如果你会需要 X Window 时，这个服务要启动。

没错！不要怀疑！只要这些就可以啦！这几个服务是必须要启动的！ 至于其它服务则都先不用启动！例如 sendmail 啦！其它林林总总的数据，都先摆着！ 我们会在后续的章节当中提到如何启动这些服务的啦！

安全性的考虑

我们的 Linux distribution 很好心的帮使用者想到很多了，所以在一安装完毕之后， 系统会开启一堆有的没有的网络服务，例如那个 portmap 之类的咚咚，以及网络打印机的 cups 服务等等， 这些东西你或许知道或许不知道，不过他就是有开启～但我们的主机明明就是用来做为服务器的， 所以这些本来预计要给 client 使用的服务其实有点『多此一举』的感觉～ 所以啦，请你将他关闭吧！就利用 ntsysv 或 chkconfig 来关闭他！ 只留下前一节咱们建议的那些服务就好了～其它的以后再说啊！

不过要记得， ntsysv 及 chkconfig 都是在管理开机是否启动某些服务的 script 而已， 所以使用 chkconfig 管理完毕后，请记得最好使用 reboot 来完整的重新加载这些服务， 然后以『 netstat -tunpl 』来看看是否有什么其它的网络服务在启动啊？ 如果有的话，在一样一样的将他关闭吧！ ^_^

课后练习

- 如何观察您 Linux 主机上面已经有多少 port 被打开了？

  1. 如果是 Linux 这个操作系统上面的话，可以利用『 netstat -tunlp 』观察已经在监听的 port 与服务的对应；
  2. 如果是想要查阅所有的 port (包含已建立的联机)，可以使用『 netstat -tunp 』来查阅；
  3. 如果不在 Linux 本机上，可以用『 nmap IP 』来处理啊！

- 如何观察程序？

  利用『 ps -aux 』或『 top 』都可以，另外，『 pstree -p 』则可以了解所有的程序相依性，而『 lsof 』 则可以察看所有程序所开的档案喔！

- 请问 LISTEN 的 port 与 daemon 的关系为何？

  正在 LISTEN 当中的埠口均是由某些服务(daemons)所启动的，所以要启动埠口就得启用某个服务， 要了解某个埠口是由那个 daemon 所启动的，就利用 netstat -tulp 来查阅。

- 请问 stand alone 与 super daemon 各是什么？

  Linux 系统的服务有独立启动(stand alone)及超级服务员(super daemon)两种启动的方式。挂在 super daemon 底下的服务可以经由 super daemon 的控管，以加强一些安全功能，不过由于还要经过 super daemon 的管理，所以服务的连接速度上会比 stand alone 慢一点。

- 请问您的 Linux 主机（不论是那个 distributions ）有关 daemon 启动与关闭的 scripts 与档案放置在那个目录下？

  各个 daemons 的启动与关闭的 scripts 是放置在 /etc/init.d/ 内， Red Hat 系统则是放到 /etc/rc.d/init.d 里面，至于 super daemon 的控管参数档案则在 /etc/xinetd.d 里面！

- 为什么阻断式服务 (DDoS) 会造成系统的当机与网络瘫痪？试由三向交握的角度来探讨。

  所谓的阻断式服务是利用三向交握程序的漏洞，多个 cient 端持续发送 tcp 封包的联机要求，但却不理会 server 端的 SYN/ACK 的封包，导致 server 端会持续启动很多的 port 在等待 client 端的回应， 那我们知道一般 port 有 65536 个，万一用完了，那系统网络就瘫痪了！所以 DDoS 会造成系统网络瘫痪的问题。 另外，由于多个 client 同时要求，所以网络频宽也会被用光！

主机基本安全之二： Linux 在线自动升级

最近更新日期：2006/09/10

在现在的 Internet 上面，Cracker 实在是太多了！这些 Cracker 会利用已经存在的系统漏洞，来进行侦测、入侵您的主机， 因此，除了未来架设防火墙之外，最重要的 Linux 日常管理工作，莫过于套件的升级了！不过，经由每日观察网络安全通报所告知的套件漏洞，以及等待各大 distribution 针对这些漏洞来提供 RPM 档案，以使 Client 来升级的过程中，实在是有点缓慢啊！因此， 目前就有很多在线直接更新的机制出现了！有了这些在线直接更新 RPM 的手段与方法，我们系统管理员在管理主机系统上面，可就轻松的多啰！赶紧来看看吧！

为何需要进行软件升级

很多朋友在网络上面常常会这样留言： 『大家好，我的 Linux 好像怪怪的，因为没有办法以 root 的身份登入了』， 呵呵！几乎可以直接告诉这位朋友：『你的系统被入侵了』！嗄～真假？早上才装好，下午被入侵？ 没错啊～但～对方是如何办到的呢？

在前一章我们不是有提到『网络联机的 port 其实是由软件所开启的』， 所以，如果该软件本身就有问题的话，那么当然你的系统就容易被攻破了！ 咦！那自由软件干嘛开发出有问题的程序啊？这是因为程序是人写出来的， 在设计之初有些奇怪的用法可能没有考虑到，或者是某些安全问题没有考虑到， 而造成程序的疏失。当这样的程序发布后，很多人会针对这些程序进行检验，如果发现问题就会回报给社群。 那么当回报后直到程序更新之前，总会有一段空窗期，这期间可能就会有些 cracker 开发出具有攻击码的程序，如果这些攻击程序散布出来的话，那么随便一个小朋友拿到这样的程序，就能攻击你啦！

这个问题并非仅存在于某个单一操作系统，而是所有的操作系统都存在这样的问题， 而且套件的漏洞倒不是一定是会被利用来进行入侵，有的时候，某些套件的漏洞可能导致您 Linux 主机的运行不良或者是容易产生系统当机等等的问题呢！所以，一个好的 Linux 主机， 他的套件最好是随时保持在较新的版本上面，这样还是比较好一点的啦！关于安全漏洞的通报您可以参考底下的网站：

- 台湾计算机危机处理小组(TWCERT)：http://www.cert.org.tw/

- Red Hat 的官方说明：http://www.redhat.com/apps/support/errata/

所以啊，并不是有防火墙就万事 OK ！你还必须要更新你的软件才行， 通常建议你，安装 Linux 完毕后的第一项工作那就是..... 立刻进行整体软件的升级！

> Tips:
> 事实上，自由软件的安全性还是比较好的！因为有太多人帮忙检验程序代码与更新程序代码， 并且，万一程序真的有问题，在发现问题到推出修补程序的期间是比较短的， 也就是说，攻击者可以利用的时间相对缩短，当然使用自由软件的我们就比较安全啦！ ^_^



如何进行软件升级

还记得你是如何安装你 Linux 上面的软件吗？不就是 rpm, tarball 与 dpkg 吗？ 所以啰，你的软件如果想要升级，那就得依据当时你安装该软件的方式来进行升级啊！ 而每种方式都有其适用性：

- RPM：这是目前最常见于 Linux distribution 当中的套件安装管理方式， 包括 CentOS / Fedora / SuSE / Red Hat / Mandriva 等等，都是使用这个方式来管理的；

- Tarball：利用软件的官方网站所释出的原始码在您的系统上面编译与安装， 一般来说，由于软件是直接在自己的机器上面编译的，所以效能会比较好一些。 不过，升级的时候就比较麻烦，因为又得要下载新的原始码并且重新编译一次。 这种安装模式常见于某些特殊软件 (没有包含在 distribution 当中)，或者是 Gentoo 这个强调效能的 distribution；

- dpkg：是 debian 这个 distribution 所使用的套件管理方式，与 RPM 很类似，都是透过预先编译的处理，可以让 end user 直接使用来升级与安装。

举例来说，如果你的系统是 CentOS ，我们知道他使用的是 RPM 类型的套件管理模式，那如果你想要安装 B2D 的软件怎办？要注意， B2D 是使用 debian 的 dpkg 来管理套件的，两者并不相同啊！要互相安装太难了！ 所以说，要升级的话，得先了解到你系统上的套件安装与管理的方法才行。

不过，有个特殊案例，那就是旧版本的 Linux (例如 Red Hat 9) 的软件升级该如何是好？ 由于旧版本的软件支持度本来就比较差，商业公司或者是社群也没有这么多心力放在旧版本的支持上， 所以，你这个时候可以选择： (1)升级到较新的版本，例如 CentOS 4.3 或者是 SuSE 10 等等，或者是 (2)利用 Tarball 来自行升级核心与软件 。不过，比较建议升级到新版本啦，因为要自行以手动方式由 Tarball 安装到最新的版本，实在是很费时费力，而且还得要常常查阅官方网站所推出的最新消息， 漏过一则都可能发生无法预期的状况。

但不管怎么说，单纯使用 RPM / Tarball / dpkg 的方式来安装与升级软件时， 你都必须要由原版光盘或者是由官方网站下载可安装的套件档案，然后再手动来实际安装到你的系统上。 如此一来，你还是得要盯着官方网站提供的信息，才能够在第一时间进行升级的动作。 唉！怎么这么麻烦？

我们都晓得在 Windows 的环境下，他有提供一个 Live update 的项目可以自动的在线升级， 甚至很多的防毒软件与防木马软件也都有推出实时的在线更新，如此一来可以让您的软件维持在最新版的状况， 真是

好啊！咦！那我们的 Linux 是否有这样的功能？如果有的话，那么系统自动进行软件升级， 不就可以轻松又快乐了？没错！确实是这样的！所以就让我们来谈一谈 Linux 的在线升级机制吧！

各种 distributions 的自动升级机制

在 Linux 最常见的套件安装方式： RPM / Tarball / dpkg 当中，Tarball 由于取得的是原始码， 所以要用 Tarball 来作在线自动更新是不太可能进行的，所以仅能用 RPM 或 dpkg 这两种套件管理的方式来进行在线更新了。

但 RPM 与 dpkg 不是有所谓的相依属性吗？ 这倒不需要担心呐!因为我们的 RPM 与 dpkg 套件档案都有一些套件的基本信息， 并同时记录了套件的相依属性 (记得使用 rpm -q 的查询吗)， 所以当分析这些基本信息并使用一些机制将这些相依信息记录下来后， 再透过一些额外的网络功能，就能够自动的分析你的系统与修补套件之间的差异， 并可进一步帮你分析所需要升级与相依属性的套件，就可达成自动升级的理想啦！

由于各家 distributions 在管理系统上都有自己独特的想法，所以在分析 RPM 或 dpkg 套件与方式上面就有所不同， 也就有底下这些不同的在线升级机制啦：

- yum：
  CentOS 与 Fedora 所常用的自动升级机制，透过 FTP 或 WWW 来进行在线升级以及在线直接安装套件；

- up2date：
  这是 Red Hat 所使用的自动升级机制，需要注册才能使用，并且依据付费与否而管制其流量；

- apt：
  最早由 debian 这个 distribution 所发展，现在 B2D 也是使用 apt ，同时由于 apt 的可移植性， 所以只要你的 RPM 可以使用 apt 来管理的话，就可以自行建立 apt 服务器来提供其它使用者进行在线安装与升级。

- you：
  所谓的 Yast Online Update (YOU) 是由 SuSE 所自行开发出来的在线安装升级方式， 经过注册取得一组账号密码后，就能够使用 you 的机制来进行在线升级。不过如果是免费的版本， 则仅有 60 天的试用期!

- urpmi：
  这个则是 Mandriva 所提供的在线升级机制!

讲了这些升级机制并且与 distribution 作了对应，你就该了解到：『每个 distribution 可以使用的在线升级机制都不相同』的啊！所以请参考你的 distribution 所提供的文件来进行在线升级的设定喔！否则就得要自行手动下载安装了！ @_@

底下鸟哥以 CentOS 4.3 提供的 yum 在线升级架构来进行说明，同时亦简单的介绍一下 B2D 这个 distribution 的 APT 来说明说明！ ^_^！那为什么不选择其它的版本来介绍呢？ 就如同局域网络那个章节里面的 distribution 选择提到的， 越稳定的版本就不容易发生程序臭虫，不断升级的情况就比较不会

发生，所以，鸟哥这里再次的说明一下， 如果要做为主机服务器之用的话，尽量选择较稳定且支持较多的版本，例如 CentOS, SuSE, Red Hat, B2D 及 debian 等 distribution 啊！

CentOS 的 yum 自动升级

我们知道 CentOS 主要是以 RPM 来作为套件的管理机制，那么 RPM 本身就有一些表头数据记录了这个套件本身的信息，包括了相依属性之类的讯息等等，yum 这个咚咚就是藉由分析这些 RPM 套件的表头数据，并且将这些表头数据事先记录下来， 当使用者要求升级或者是安装的时候， yum 就会透过分析这些表头数据来决定下载的档案， 这些下载的档案当然包括了相依属性的套件了，所以说，yum 已经主动克服了套件之间的属性相依问题啰！很棒吧！

那么 yum 是如何动作的呢？基本上是这样的：

- 先由设定档判断 yum server 所在处；
- 连接到 yum server 后，先下载新的 RPM 档案的表头数据；
- 分析比较使用者所欲安装/升级的档案，并提供使用者确认；
- 下载使用者选择的档案到系统中的 /var/cache/yum ，并进行实际安装；

所以说，找到合适的 yum server 是挺重要的一件事啊！

---

yum 的设定档

基本上，在你一安装完 CentOS 之后，系统就主动的帮你建立好 CentOS 的 yum server 设定了， 他的设定档在：

- /etc/yum.conf
- /etc/yum.repos.d/CentOS-Base.repo

其中，那个 yum.conf 是主要设定档，可以设定一些环境参数之类的，至于 CentOS-Base.repo 则是主要的 yum server 选择的数据，你可以直接修改 CentOS-Base.repo 这个档案即可。 另外，台湾地区的 CentOS 镜像站台 (mirror) 可以选择义守大学的 FTP 网站， 例如底下的连结：

- http://ftp.isu.edu.tw/pub/Linux/CentOS/

截至目前为止 (2006/09/xx)，最新的 CentOS 是 4.4 版，所以上头这个连结你可以进入 4.4 那个目录，就能够看到很多 CentOS 提供的各项套件数据了。其中比较重要的两个目录是：『 o s』以及『 update 』，分别是基础套件以及修补过后的套件啦！既然知道了台湾地区的 FTP 网站后，自然就不需要连接到美国去下载档案， 那么联机下载的速度当然就会比较快啦！不过，你就得要自行修改修改设定档了！

不过 CentOS 官方网站则是建议使用国码来作为镜相网站的选择依据， 如此一来在大版本相同的环境下 (4.3 -> 4.4) 咱们的 CentOS 是可以自动升级到不同版本中的！ 所以，鸟哥的设定档是改成这个样子的：

```
[root@linux ~]# vi /etc/yum.repos.d/CentOS-Base.repo
[base]
```

```
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch
&repo=os&cc=tw
# 注意！上面两行是同一行！
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-centos4

#released updates
[update]
name=CentOS-$releasever - Updates
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch
&repo=updates&cc=tw
# 注意！上面两行是同一行！
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-centos4
......(底下省略).....
```

主要是将 mirrorlist 那个变量的最后面加上国码『&cc=tw』就可以了！如果未来有较新的版本时， 那么你的 yum 就能够自动升级啰！另外，除了 [base] 与 [update] 之外，其实 CentOS 还提供很多的额外套件， 这包括了： addons, extras, centosplus, contrib 等等，这些数据你也可以加入到设定档当中， 来帮助你容易安装某些非正规支持的套件数据啊！ ^_^

另外最要注意的是，在设定档当中所指定的镜像站台 (mirror) 离你越近越好，而且频宽越大越好， 这样你就可以比较方便快速的下载啊！而且，你必须已经成功的连结到该镜像站台才行， 否则在执行 yum 时会发生某些问题喔！另外，你必须是 root 的身份才能使用 yum 啊！ 也就是说：

- 你必须使用 root 的身份来执行 yum ；
- 设定档内指定的镜像站台必须能与你进行网络连接；
- 镜像站台频宽越大越好，所以选择离你越近的镜像站越好！

---

yum 的安装、升级、移除、查询等功能

yum 可不止能够在线自动升级而已，他还可以作查询、套件群组的安装、整体版本的升级等等，好用的哩！先来谈论一下 yum 这个 client 端的指令用法吧：

```
[root@linux ~]# yum [option] [工作项目] [套件]
参数：
option：主要的参数，包括有：
   -y ：当 yum 询问使用者的意见时，主动回答 yes 而不需要由键盘输入；
   --installroot=/some/path ：安装在其它的路径，而不在目前目录树的架构中；
        对于建立虚拟机器相当有帮助！不过，一般使用者应该用不到。
[工作项目]：由于不同的使用条件，而有一些选择的项目，包括：
   install ：指定安装的套件名称，所以后面需接『 套件名称 』
   update  ：进行整体升级的行为；当然也可以接某个套件，仅升级一个套件；
```

```
   remove   ：移除某个套件，后面需接套件名称；
   search   ：搜寻某个套件或者是重要关键字；
   list     ：列出目前 yum 所管理的所有的套件名称与版本，有点类似 rpm -qa；
   info     ：同上，不过有点类似 rpm -qai 的执行结果；
   clean    ：下载的档案被放到 /var/cache/yum ，可使用 clean 将他移除，
              可清除的项目：packages | headers | metadata | cache 等；
另外，在[工作项目]部分还可以具有整个群组套件的安装方式，如下所示：
   grouplist    ：列出所有可使用的『套件组』，例如 Development Tools 之类的；
   groupinfo    ：后面接 group_name，则可了解该 group 内含的所有套件名；
   groupinstall：这个好用！可以安装一整组的套件群组，相当的不错用！
                 更常与 --installroot=/some/path 共享来安装新系统
   groupupdate ：升级整个套件群组；
   groupremove ：移除某个套件群组；

范例一：搜寻 CentOS 的更新主机上是否有 RAID 磁盘阵列相关套件？
[root@linux ~]# yum search raid
.....前面省略.....
mdadm.i386                              1.6.0-3                base
Matched from:
mdadm controls Linux md devices (software RAID arrays)
mdadm is used to create, manage, and monitor Linux MD (software RAID)
.....后面省略.....
# 看到否？输出数据的特殊字体那一行就显示了你可以安装的套件名称然后你可以这样：

[root@linux ~]# yum info mdadm
Name    : mdadm
Arch    : i386
Version: 1.6.0
Release: 3
Size    : 84 k
Repo    : base
Summary: mdadm controls Linux md devices (software RAID arrays)
Description:
.....后面省略.....
# 瞧一瞧啊！套件的版本名称、数据大小、还有该套件出处 (base)！
# 需要注意看的是 Summary 与 Description 这两个注意事项内容！
```

yum 真是个很好用的东西，他可以直接查询是否有某些特殊的套件， 你可以利用『 yum search "一些关键词" 』或者是『 yum list 』列出所有的套件名称， 然后再以正规表示法取得关键词，或者是『 yum list "套件名称" 』 就能够知道该套件的用途， 最后再决定要不要安装啊！上面的范例一就是在找出磁盘阵列的管理软件， 如果确定要安装时，那就可以这样处理：

```
范例二：安装某个套件吧！以 mdadm 为例：
[root@linux ~]# rpm -q mdadm
package mdadm is not installed
```

```
# 鸟哥的主机并没有安装这个玩意儿～所以底下开始安装先！

[root@linux ~]# yum install mdadm
Setting up Install Process
Setting up repositories
update                    100% |=========================|  951 B    00:00
base                      100% |=========================| 1.1 kB    00:00
addons                    100% |=========================|  951 B    00:00
extras                    100% |=========================| 1.1 kB    00:00
# 上面这个阶段在读取 RPM 档案的文件头数据；

--> Populating transaction set with selected packages. Please wait.
---> Downloading header for mdadm to pack into transaction set.
mdadm-1.6.0-3.i386.rpm    100% |=========================| 8.2 kB    00:00
---> Package mdadm.i386 0:1.6.0-3 set to be updated
--> Running transaction check
# 上面这个阶段则是在下载档案以及准备更新的阶段

Dependencies Resolved
=============================================================================
 Package                 Arch       Version          Repository        Size
=============================================================================
Installing:
 mdadm                   i386       1.6.0-3          base               84 k

Transaction Summary
=============================================================================
Install      1 Package(s)
Update       0 Package(s)
Remove       0 Package(s)
Total download size: 84 k
Is this ok [y/N]: y
# 至于这个阶段则在分析相依属性，并且让使用者确认下载开始

wnloading Packages:
(1/1): mdadm-1.6.0-3.i386 100% |=========================|  84 kB    00:00
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing: mdadm                        ######################### [1/1]

Installed: mdadm.i386 0:1.6.0-3
```

```
Complete!
# 最终则下载与安装的结果!
```

瞧!经过 yum 我们可以很轻松的就安装好一个软件,并且这个软件已经主动的帮我们做好相依属性的克服了, 真是方便到爆!另外,你必须要知道,刚刚那个被下载安装的 mdadm 档案被放置到 /var/cache/yum 里面去了, 如果你要节省硬盘空间的话那么可以在安装完毕后将该档案移除,就用:

```
[root@linux ~]# yum clean packages
Cleaning up Packages
2 packages removed
```

这样就能够清除掉已下载的档案啰～节省一下硬盘空间啊!OK～那如何进行整体的更新呢? 比如说鸟哥刚刚装完了 CentOS 4.3 ,但这个版本已经推出若干时间,所以也已经作了很多更新了, 那鸟哥如何整体更新啊?很简单,就用如下的指令:

```
[root@linux ~]# yum -y update
```

加一个『 -y 』的参数可以让系统自动帮你回答『 yes 』,在背景处理时会比较方便一点。 如果你是第一次执行,那就会发现:哇!怎么下载的数据量到达数百 MB 之谱!没错啊! 所以记得 /var/ 这个目录的容量要给大一点才行! 否则就会出现无法完整下载所有更新档案的问题啊! @_@

- 安装套件群组的功能

什么是『套件群组』呢?还记得在安装的时候有出现套件选择的地方吧? 在那个时候你选择的数据可不是『套件名称』喔,而是一堆『套件群组』,举例来说, 你会看到『KDE 桌面环境』之类的,而不是每个 KDE 桌面的各项套件名称,对吧!那个咚咚就是『套件群组』啦! 由于各大 distributions 预设都没有选择发展工具 (Development Tools), 这些工具包含了 gcc, kernel-devel 等等,那么你如何使用 yum 一口气安装呢? 看看底下的范例:

```
范例三:查询与安装『套件群组』
[root@linux ~]# yum grouplist
Installed Groups:
   Administration Tools
   Authoring and Publishing
   Compatibility Arch Support
.....中间省略.....
Available Groups:
   Development Tools
   XFCE-4.2
.....中间省略.....
Done
# 看到没!上面就列出来你已经安装的套件群组,还有尚可安装的其它套件群组,
# 真是非常的方便!那么如何知道 Development Tools 里面有啥咚咚?

[root@linux ~]# yum groupinfo "Development Tools"
Group: Development Tools
 Required Groups:         <==所需要的相依属性数据
```

```
   Development Libraries
 Default Metapkgs:        <==预设内部所需要的中继套件
   Emacs
 Optional Metapkgs:       <==最好还含有这些套件较佳
   Ruby
   XEmacs
.....中间省略.....
 Mandatory Packages:      <==一些所需要的套件数据
   pkgconfig
   gcc-ppc32
   make
   gcc
   autoconf
.....中间省略.....
 Default Packages:
   gcc-g77
   cscope
.....中间省略.....
 Optional Packages
   dejagnu
   ElectricFence
   gcc-gnat
.....中间省略.....
# 总共会列出来这个『套件群组』内含有的各项资料，如果你需要安装的话，就可以：

[root@linux ~]# yum groupinstall "Development Tools"
```

利用这个『 yum groupinstall "套件群组名" 』可以让你一口气安装很多的套件， 而不必担心某个套件忘记装了！实在是很不错啦～而且利用 groupinfo 的功能你也可以发现一些不错的套件数据， 如此一来，你就可以更方便的管理你的 Linux 系统了，很不错吧！

例题：请设定一下工作排程，让你的 centOS 可以每天自动更新系统

答：

可以使用『 crontab -e 』来动作，也可以编辑『 vi /etc/crontab 』来动作，由于这个更新是系统方面的，所以鸟哥习惯使用 vi /etc/crontab 来进行指令的说明。其实内容很简单：

40 5 * * * root yum -y update && yum clean packages

这样就可以自动更新了，时间订在每天的凌晨 5:40 ，并且更新完成后会主动的将下载的套件数据移除喔！

不同版本间的升级

什么！不同的版本之间可以直接『网络』升级了喔？没错！而且整个流程还挺简单的， 升级完成之后，绝大部分的服务都还不会有困扰！真是很不错啊！那什么是『不同版本？』举例来说， CentOS 4.2 升级到 CentOS 4.3 算是一种，而 Fedora Core 1 升级到 CentOS 4.3 则又是另外一种， 同样是 CentOS 的升级比较容易，尤其 4.3 本来就是架构在 4.2 上面持续发展的结果； 不过如果是 Fedora Core 的话，可能就比较麻烦一点点。底下我们分别谈一谈这两种方式的升级吧！

- CentOS 4.2 升级到 CentOS 4.3

在 CentOS 的发展理念当中，如果推出了 4.3 ，那么 4.2 以前的 4.x 版本就不会继续发展， 所以使用者必须要将原本的 4.2 直接提升到 4.3 才行。那么需要作些什么动作呢？ 不需要啊！只要修改一下 yum 的设定档就好了。首先，同样需要找到最近的镜像站台， 我们依旧以义守大学的 FTP 网站来提供所需要的套件数据，修改成这样：

```
1. 先修改 /etc/yum.conf
[root@linux ~]# vi /etc/yum.conf
.....前面省略.....
# 直接在档案的最底下加入这一行来增加一些额外的功能：
plugins=1

2. 再修改 /etc/yum.repos.d/CentOS-Base.repo
[root@linux ~]# vi /etc/yum.repos.d/CentOS-Base.repo
# 内容与『 yum 的设定档』说明相同，请回到本小节的最前面查阅该设定
```

因为 /etc/yum.repos.d/CentOS-Base.repo 的内容与前面相同， 所以鸟哥在这里不再浪费篇幅，请往前翻阅吧！设定好了之后，接下来给他进行：

```
[root@linux ~]# yum upgrade
```

记得是『 upgrade 』而不是『 update 』喔！两者用法不同啊！ 然后接下来就是一段时间的等待啊！没办法，因为从网络上面捉数据下来是需要时间的！ 还好鸟哥的环境是在学术网络内，所以连结同样是学术网络的义守大学还挺快的就是了！ ^_^！ 整个升级的时间大约花费 20 分钟以内，升级完毕之后，重新开机瞧一瞧登入画面！哇！变成 CentOS 4.3 了， 真是快速又方便！而且原本有启动的服务几乎没有任何问题，同样可以正常的启动呐！^_^

上面的动作你可以参考底下这一篇官方说明文件：

- http://www.centos.org/modules/news/article.php?storyid=118

- Fedora Core Release 1 升级到 CentOS 4.3

如果你使用的是旧版的 Linux distributions ，例如 Fedora core release 1, Red Hat 9 等等的系统，这些系统已经旧到没有什么更新的软件出来，所以如果套件有臭虫而需要更新时， 你可能就得要使用 Tarball 的方式手动的给他『 configure, make, make install 』等等的， 好累啊～那如果我安装 CentOS 呢？如果需要主动重新安装的话，那旧的数据不是会不见吗？ 又得要备份，重新处理等等的，还是很累啊！

没关系！有 yum 就搞定了！你可以将你的 FC1 升级到 CentOS 4.3 了，而且是『在线更新』喔！ 厉害吧！不过，因为 FC1 使用的核心是 2.4 版，但 CentOS 4.3 使用的是新的 2.6.x ， 这两种核心可不能互相

更新啊！所以啰，我们还需要一些额外的动作来进行升级，而不像前面 CentOS 4.2 升级到 4.3 那么简单！鸟哥底下的动作是参考这几篇：

- twu2 兄提供的不同版本间升级：http://phorum.study-area.org/viewtopic.php?t=28648
- CentOS 官方网站提供的一些升级建议：
  http://www.centos.org/modules/newbb/viewtopic.php?topic_id=428&forum=6
  http://www.centos.org/modules/newbb/viewtopic.php?topic_id=382

鸟哥底下以 FC1 为例来进行整个升级的动作，不过你得要了解的是，每个人的 Linux 都不相同， 因此虽然鸟哥实作是成功的，不过不代表你的环境一定会成功， 所以，重点是....『请做好备份！』 以免升级不成功时，导致整个数据的损毁，那就得不偿失了！

```
0. 前处理：先准备好你的数据，以及删除不需要的数据
[root@linux ~]# yum clean packages headers
# 先删除原本的 yum  数据，因为 FC1 使用的也是 yum ，
# 所以最好先将之前 FC1 的 yum 数据删除，比较不会有问题；
```

在这个步骤当中，你最好先将一些重要数据备份起来，包括 /etc 与 /home 整个目录， 还有其它你有开启的服务的数据，包括 MySQL 或者是 WWW 的网页数据等，请自行备份喔。 另外，我们的 yum 预设是将下载的套件通通放置到 /var/cache/yum 当中，所以你的 /var 目录所在 partition 的容量也需要至少 1GB 以上的容量，而且安装软件所在目录 /usr 所在的 partition 最好也能够有 2GB 以上的空间，否则容易出现空间不足的错误讯息而无法继续。

那如果真的空间不足怎么办？你可以将一些不需要的套件先移除啊！举例来说， 你可以利用『 yum groupremove "Development Tools" 』之类的指令先将这些不是必备的套件群组移除， 以及其它 X 相关的套件也可以先移除，等到升级完毕后再以 yum 来重新安装即可， 这样可以节省很多升级时分析相依属性所花费掉的时间喔！

同时你得要确认你的镜像站台，以及 CentOS 的数字签章档案已经安装到 RPM 数据库当中才行， 同样的，鸟哥还是以义守大学的 FTP 站为主要的来源镜像站，但是你必须要先取得一些 RPM 相关的档案， 这些档案由于会与 FC1 原本的套件产生不相符合的特性，所以需要先捉下来并且强制安装才行！ 请你自行连上：

- http://ftp.isu.edu.tw/pub/Linux/CentOS/

然后选择最新的版本，例如鸟哥使用 4.3 (2006/08/10 以前) 这个版本，所以直接点选他， 然后依序选择『os』->『i386』->『CentOS』->『RPMS』，然后下载底下这些档案 (注：套件的版本号码可能会有些不同喔！)：

- centos-release-4-3.2.i386.rpm
- centos-yumconf-4-4.5.noarch.rpm
- kernel-2.6.9-34.EL.i686.rpm
- udev-039-10.12.EL4.i386.rpm

假设我的这些档案捉下来后放到 /root 下，那接下来的动作是：

```
1. 安装升级所需要的 RPM 档案：
```

```
[root@linux ~]# rpm --import  \
> http://ftp.isu.edu.tw/pub/Linux/CentOS/4.3/os/i386/RPM-GPG-KEY-centos4
[root@linux ~]# rpm -Uvh centos-release-4-3.2.i386.rpm
[root@linux ~]# rpm -Uvh centos-yumconf-4-4.5.noarch.rpm
[root@linux ~]# rpm -ivh --force --nodeps kernel-2.6.9-34.EL.i686.rpm
# 在这个动作时，由于我们的核心并非是 CentOS ，所以这个动作会发生一些错误，
# 先不要理他，待会儿的动作再重新处理即可。
[root@linux ~]# rpm -ivh --force --nodeps udev-039-10.12.EL4.i386.rpm
```

由于 kernel 2.6 使用的装置管理是以 udev 这个套件来处理的，与原本的 kernel 2.4 并不相同， 为了避免使用者进行错误的安装，所以 yum 会主动的分析核心与装置管理套件的差异， 如果两者无法配合就产生错误讯息且强制中断 yum ，那结果就是导致无法以 yum 进行升级啦！ 要躲过这个困扰，你就得要先手动的安装上头那几个 RPM 档案。同时安装 kernel 的过程当中会发生错误讯息， 先不要理他，等到后续步骤再来重新处理即可！ 既然 kernel 2.6 会与 2.4 版冲突，而鸟哥的原本的 FC1 本来就有 2.4 版的核心嘛！ 那即使安装了新的 kernel-2.6.9-34.EL.i686.rpm 核心，但核心版本冲突的问题还是没有解决的， 所以啊，你就得要这样做了：

```
2. 移除会有冲突的 2.4 版核心，连同其原始码
[root@linux ~]# rpm -qa | grep kernel
kernel-2.4.22-1.2197.nptl
kernel-2.4.22-1.2115.nptl
kernel-2.4.22-1.2199.nptl
kernel-source-2.4.22-1.2197.nptl
kernel-source-2.4.22-1.2199.nptl
.....其它省略.....
[root@linux ~]# rpm -e kernel-source-2.4.22-1.2199.nptl
[root@linux ~]# rpm -e kernel-source-2.4.22-1.2197.nptl
[root@linux ~]# rpm -e kernel-2.4.22-1.2115.nptl
[root@linux ~]# rpm -e kernel-2.4.22-1.2197.nptl
[root@linux ~]# rpm -e kernel-2.4.22-1.2199.nptl
# 反正就是找到核心与核心原始码的套件后，就将他移除！记得套件名为：
# kernel 与 kernel-source  ，其它的不用动！
[root@linux ~]# rpm --rebuilddb
```

将一些旧版本的核心给他移除后，你的系统就只有剩下刚刚安装的那个 CentOS 的新核心， 所以核心版本的冲突当然就不存在了。要注意的是，你现在绝不可重新开机，否则你的系统就挂了！ 因为没有核心了喔！ 切记切记！然后开始要设定好升级时要使用的 yum 设定档了！

```
3. 规范设定档
[root@linux ~]# vi /etc/yum.upgrade
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
pkgpolicy=newest
distroverpkg=fedora-release
```

```
installonlypkgs=kernel kernel-smp kernel-hugemem kernel-unsupported
tolerant=1
exactarch=1
plugins=1

[upgrade]
name=CentOS-4.3 - upgrade
baseurl=http://ftp.isu.edu.tw/pub/Linux/CentOS/4.3/os/i386/

[update]
name=CentOS-4.3 - Updates
baseurl=http://ftp.isu.edu.tw/pub/Linux/CentOS/4.3/updates/i386/
```

设定文件里面先只要有这两个项目即可，因为我们仅注视在整体版本的升级， 所以其它的额外功能部分先不要理他！以后再来处理即可啊！接下来就准备要升级了！这样做吧：

```
4. 开始升级的动作：
[root@linux ~]# yum -y -t -c /etc/yum.upgrade upgrade
```

理论上，如果你的 FC1 没有安装什么奇怪的软件，而且『硬盘空间也足够』 ，那么 yum 应该会开始帮你一个一个的下载软件并且分析属性相依问题后，就开始进行安装的步骤。 不过，如果发现一些软件冲突的问题时，那么你就得要先以 rpm -e 将旧的软件先移除， 等到升级完毕后再安装回来即可。只不过这个动作将依你的环境而有所不同。 鸟哥的 FC1 实在是旧的可以，所以很多软件都有冲突，因此事先移除了很多的套件， 忙了快要半小时后，系统才顺利的开始进行安装。由于鸟哥主机所在环境的网络下载的速度尚可而已， 所以由开始下载到升级完毕，大概花了一个半小时左右！ @_@

```
5. 重新安装核心，并处理 RPM 数据库与更新其它套件
[root@linux ~]# rpm -ivh --force --nodeps kernel-2.6.9-34.EL.i686.rpm
[root@linux ~]# rpm --rebuilddb
[root@linux ~]# yum update
```

透过这三个指令我们可以再将刚刚没有完成的核心安装一遍，同时再以新的 yum 来进行升级， 这个时候我们的系统应该是很 OK 的啦！不过，有的小问题，那就是『开机的设定档还没有更新』喔！ 所以你还得要这样做：

```
6. 设定开机设定档
[root@linux ~]# vi /boot/grub/menu.lst
default=0
timeout=10
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title CentOS (2.6.9-34.0.2)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.9-34.0.2.EL ro root=/dev/hda1 hdd=ide-scsi rhgb
        initrd /boot/initrd-2.6.9-34.0.2.EL.img
title CentOS (2.6.9-34.EL)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.9-34.EL ro root=/dev/hda1 hdd=ide-scsi rhgb
```

```
        initrd /boot/initrd-2.6.9-34.EL.img
```

在今日 (2006/08/10) 能够安装的最新核心版本是 2.6.9-34.0.2.EL ，所以上面的第一个 title 才会是这样的设定数据，要记得与你的环境符合啊(如果你不是使用 4.3 来升级)！如果忘记上面各项数据的意义，请参考『鸟哥的 Linux 私房菜 基础篇 boot loader 』的介绍啊！ 如果设定好了这个开机信息，嘿嘿！请重新开机试看看吧！ ^_^

基本上这样已经处理完毕了！也就是说，你的系统应该是由 FC1 顺利的转成 CentOS 4.3 才对！ 不过，有些小细节你依旧需要注意喔：

- 观察您的系统是否有启动原有的服务，亦可利用 chkconfig --list；
- 如果你的旧系统曾有使用 tarball 的方式来安装软件，则升级完毕后你必须要重新安装一次该软件；
- 系统可能会产生很多的 .rpmnew 的档案，请使用 locate 搜寻，并且重新处理设定档。

毕竟之前的版本太旧了，所以 RPM 升级时会将新套件的设定档存成 *.rpmnew ， 你最好不要使用旧的设定档，而是以新的设定档来重新修改比较好！这样服务的运作应该会比较顺畅一些啊！ 到此为止，恭喜您啊！ ^_^

Debian 的 apt 自动升级：以 B2D 为例

APT 最早是 debian 这个社群的 Linux distribution 用来作为套件管理的一项机制， 后来实在很方便又好用，所以就被利用来做为其它 distributions 的在线升级管理机制了！ 所以你会在很多地方看到有 FC 系列的 APT 服务器、或者是自订的 APT 服务器等等的。 这个 APT 的使用与 yum 很类似呐！同样也是透过套件的表头分析后，然后与系统本身数据比对， 因此同样也克服了属性相依的困扰了，非常方便！底下我们就分别来谈一谈吧！

APT 的设定档

如同前面提到的 yum 一样，既然是在线升级，那么我们自然得要找到相对应的服务器啰！ 而一般有提供 apt 服务的 distributions 事实上他们都已经做好设定档内相关的服务器选择了， 例如 B2D 就是一个例子。不过， 如果你是使用类似 Red Hat 9, Fedora Core Release 等版本的话， 可能由于种种原因让你不能将该主机升级到类似 CentOS 等较新的版本时，那你就可以尝试到底下的网站找找看 APT 主机了：

- http://apt.freshrpms.net/

找到属于你的 Linux 版本后，安装 apt 就可以使用 apt 服务器所提供的套件升级机制啰！ 底下赶紧谈一谈，那么如何处理 APT 的设定档呢？ APT 的设定文件都放在 /etc/apt 目录下，而在该目录下，规范 APT 主机的设定则放到 /etc/apt/sources.list 当中。 以 B2D 为例，他的 sources.list 是这样的：

```
1. 先处理 APT 服务器来源的设定数据：
[root@linux ~]# vi /etc/apt/sources.list
deb ftp://debian.tnc.edu.tw/pub/debian/ stable main non-free contrib
deb http://security.debian.org/ sarge/updates main contrib non-free
```

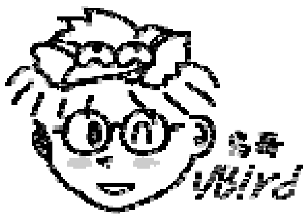
```
deb ftp://debian.tnc.edu.tw/pub1 b2d/
deb http://ftp2.de.debian.org/debian-volatile sarge/volatile main
# 上面的格式是这样的：
# <deb 的标头格式> <APT 服务器网址> <相对网址路径> <目录一> <目录二> ...
```

其实设定档的内容很简单，第一个字段指的是『用哪一种套件管理员』的意思，主要有 deb 以及 rpm 还有 rpm-src 等，deb 是 debian 专属的套件管理方式啦！ ^_^！第二栏以后就得要一起看才行！ 以上表的第一行为例，其实他是说，提供 deb 的网址有三个，分别是：

- ftp://debian.tnc.edu.tw/pub/debian/stable/main
- ftp://debian.tnc.edu.tw/pub/debian/stable/non-free
- ftp://debian.tnc.edu.tw/pub/debian/stable/contrib

这样看懂了吗？第二栏与第三栏要加在一起，那就是个完整的服务器网址！ 后面接的几个数据则是在该网址底下的数个目录，那就很容易看懂了吧！ ^_^ 应该是不难啦～如果您是使用 b2d 的话，那恭喜你！不用修改就能用 apt 啰！ ^_^

Tips:
事实上， /etc/apt/apt.conf 还可以提供其它额外的设定喔，例如使用 Proxy 以及规范下载后的套件在安装完毕后应该进行的处理步骤，举例来说，是否需要将该套件移除！ 你应该要使用 man apt.conf 查询一下该档案的用法喔！



实际使用 APT

APT 的使用也实在是很简单！只要利用 apt-get 即可！不过，不同于 yum 每次都会进行 RPM 档案表头的分析， APT 则是将套件表头的下载与实际的安装分成两个动作分别执行，先来谈一谈 apt-get 这个指令的用法吧！

```
[root@linux ~]# apt-get [-qy] [-c config_file] [更新项目] [套件名称]
参数：
-q ：不要在屏幕上输出讯息，常用在背景环境的执行当中喔！
-y ：自动在进行 apt-get 时回答 y 的响应；
-c ：后面接的是设定文件，一般系统会主动的以 /etc/apt 内的设定档为依据。
[更新项目]：要 apt-get 进行的工作，主要有这几项：
   update      ：就是更新服务器与客户端的套件表头清单，这个动作务必要进行！
   install     ：后面需要加上要安装的套件名称才行！
   upgrade     ：进行『已安装套件』的完整升级，不过未安装套件则不予安装；
   dist-upgrade：以 upgrade 相似，但是当新版本的套件有其它相依属性的套件加入时，
                 单纯的 upgrade 将无法进行安装，此时就得要使用 dist-upgrade 了！
   clean       ：清除已经下载到 /var/cache/apt/archives/ 的套件档案。
   remove      ：移除某个套件啊！

范例一：进行套件标头更新后，进行整体套件的更新动作
[root@linux ~]# apt-get update
```

```
下载:1 ftp://debian.tnc.edu.tw stable/main Packages [3349kB]
下载:2 http://ftp2.de.debian.org sarge/volatile/main Packages [3893B]
.....中间省略.....
读取 3868kB 用了 24s (159kB/s)
读取套件清单中... 完成

[root@linux ~]# apt-get dist-upgrade
读取套件清单中... 完成
了解套件依存关系中... 完成
筹划升级套件中...完成
下列的套件都将被【删除】：
  blt-common ettercap-plugins libgdbmg1-dev
下列的【新】套件都将被安装:
  dictionaries-common ettercap-common .....后面省略.....
下列的套件都将维持旧版本:
  fontconfig libxft-dev libxft2 libxft2-dbg .....后面省略.....
下列的套件都将更新:
  apache apache-common apache-utils apache2 .....后面省略.....
更新 105 个套件，新安装 32 个套件，删除 3 个套件，另不更新 7 个套件。
需要下载 122MB 的档案。
解压缩后将消耗 39.6MB 的空间。
继续执行吗？ 是按 [Y] 键，否按 [n] 键 y
下载:1 ftp://debian.tnc.edu.tw stable/main libc6-dev 2.3.2.ds1-22sarge3 [2535kB]
下载:2 http://security.debian.org sarge/updates/main login 1:4.0.3-31sarge8 [576kB]
.....中间省略.....
读取 122MB 用了 11m47s (172kB/s)
正在预先设定套件 ...
(正在读取数据库 ... 系统目前总共安装有 112550 个档案和目录。)
正预备替换 libc6-dev 2.3.2.ds1-22 (使用 .../libc6-dev_2.3.2.ds1-22sarge3_i386.deb)
正在解压缩替换的套件档 libc6-dev ...
.....中间省略.....
Please *restart* your Apache2 !
Y/N ?
y
.....中间省略.....

[root@linux ~]# apt-get clean
# 这个动作会将刚刚下载的几个 deb 的套件给他移除！节省硬盘空间！
```

请记得，那个 update 的参数并不是在进行更新， 而是在进行服务器与客户端的套件表头清单更新而已，但这个动作相当重要，如果你没有作这个动作的话，你的套件就不会更新了！ 在 apt-get update 后，再使用 apt-get dist-upgrade 这样就能够将整个系统给他升级了！很快乐吧！不过，由于我们没有加上『-y』这个参数， 所以在上表当中，我们会老是需要输入一些有的没有的指令， 这样的话，就不适合作为背景的自动升级了！所以，如果你想要在背景以 crontab 的方法自动的帮你升级的话，在 B2D 的环境下使用：

```
[root@linux ~]# vi /etc/crontab
40 5 * * * root /usr/bin/apt-get update && /usr/bin/apt-get -y dist-upgrade
```

这样每天的凌晨 5:40 就会自动的进行整体升级，而且会主动的克服相依属性的问题喔！ 另外，除了完整的将套件给他全部升级之外，我们还可以利用 apt 服务器的功能来进行查询的动作喔！ 这个时候就得要 apt-cache 来帮忙了！

```
[root@linux ~]# apt-cache [搜寻项目]
参数：
[搜寻项目]：apt-cache 可以搜寻 apt 所列出的套件标头数据喔！可用项目有：
pkgnames：列出本系统上面的所有套件名称！！有点类似 (rpm -qa)；
dump    ：列出所有的套件标头以及其相关的相依属性套件！
search  ：后面可接要搜寻的字符串，例如 apt-cache search postfix
show    ：后面接套件名称，可以显示出该套件的主要内容的描述！
showpkg ：列出后面所接套件的相依属性以该其套件提供的相关功能！
depends ：可以列出与后面所接套件有相依属性或者是冲突的相关数据！

范例一：找出与 grep  有关的套件
[root@linux ~]# apt-cache search grep
.....前面省略.....
grep - GNU grep, egrep and fgrep
grep-dctrl - Grep Debian package information
.....后面省略.....

[root@linux ~]# apt-cache show grep
Package: grep
Essential: yes
Priority: required
Section: base
Installed-Size: 660
Maintainer: Ryan M. Golbeck <rmgolbeck@debian.org>
Architecture: i386
Version: 2.5.1.ds1-4
Provides: rgrep
Pre-Depends: libc6 (>= 2.3.2.ds1-4)
Conflicts: rgrep
Filename: pool/main/g/grep/grep_2.5.1.ds1-4_i386.deb
Size: 170290
MD5sum: 68196ad14b098b0eb4b91f4a7cfa8ff2
Description: GNU grep, egrep and fgrep
 'grep' is a utility to search for text in files; it can be used from the
 command line or in scripts.  Even if you don't want to use it, other packages
 on your system probably will.

[root@linux ~]# apt-cache depends grep
```

```
grep
  特别依存关系: libc6
  冲突: <rgrep>
```

瞧！利用 apt-cache 就能够找到很多有用的信息！包括利用 show 这个参数也能够将该套件重要的项目给他列出来！ 以上面的 grep 这个套件为例， apt-cache 就列出很多例如版本信息、冲突信息 (conflicts) 等等， 尤其是描述 (Description) 的部分，就更可以让使用者了解该套件的用途了！ ^_^！另外， 那个 depends 则可以特别列出与该套件有冲突或者是相依属性的文件名称！也是个很有帮助的参数喔！ 至于如果你想要安装一个套件的话，例如 zlibc 这个套件时，就可以这样做：

```
[root@linux ~]# apt-get install zlibc
```

如同前面提到的，如果你原本的系统并不是使用 apt 来进行在线升级的机制，而你想要使用 apt 的话，目前很多服务器都有提供相对应版本的升级，其中以 RPM 套件管理的 Red Hat 与 Fedora 最常见！ 台湾杨锦昌老师也提供了完整的 APT 教学，您可以看看：

- http://163.19.59.1/~linux/student_samba/apt/apt_server.html

当然，还是那句老话，除非您的主机上面有专属的软件需要该版本的 distribution 才能执行， 当转成其它版本可能会发生无法执行的困扰时，那你只好使用旧版的 distribution， 并且找到对应的 APT 或 yum 服务器，或者是『你自己建立一个 APT/yum 服务器』来提供自己升级！ 比较能够免除一些程序臭虫的困扰。如果没有以上的困扰，那就直接升级到比较新的版本吧！ 『比较新的版本不一定比较好，不过，在大部分的情况下， 较新版本对硬件的支持以及安全性方面，都会比较好一些。』但是，如果你的主机明明运作的很好， 短期你也不需要什么新硬件的增加，而且你自己有在注意各个套件的安全性时，那不需要升级也是没有问题的啦！

重点回顾

- 由于程序是由人所撰写的，因此程序在执行的过程中难免可能会出现一些安全性的问题或者是程序臭虫的问题。 所以，绝大部分的情况下，将套件保持在最新的版本较能够避免被 cracker 所入侵的问题
- 绝大部分的自由软件之维护的人员比专属软件还要多，所以程序发生臭虫后的除错与更新时间较快速！！
- 你不一定要将你的 distribution 更新到最新，不过，更新到最新可以保有较佳的硬件支持与网络安全。
- 由于 distribution 发展的不同，基本的套件安装可以分为 rpm, dpkg 及 tarball 三种常见模式；
- 各个 distribution 均有推出自家的在线升级机制，如 CentOS 的 yum, Debian 的 apt, Red Hat 的 up2date 以及 SuSE 的 YOU 等等。各种版本均不可混用。
- yum 这个升级方案在使用时，会主动的 (1)下载表头数据与 (2)进行使用者所需要的更新动作；
- apt-get 必须要使用 apt-get update 更新表头数据后，才能够进行使用者所要求的动作！
- 在线升级机制常常需要使用到 crontab 的工作排程支持；

课后练习

- 请找到您的 distribution 所提供的在线升级机制，立刻进行全部更新的动作。
- 请前往台湾计算机危机处理小组注册，并取得各项危险通告的电子邮件通知！
- 在 RPM base 的系统当中，如果升级的套件所含有的设定文件在系统当中已经被更动过，则该设定档会如何被安装到系统中？

  如果设定档已经被改过，则更新的设定档会被储存成为 *.rpmnew 的扩展名。 并建议使用者应该要将旧的设定档备份，然后以新的设定档来进行重新设定。对于该软件的执行稳定性会较佳。

- 承上题，如何找出系统上面含有被更动过的设定档？以 rpm base 的系统为例？

  你可以利用『 locate rpmnew 』来找出已经安装的新版本设定档，并据以处理该数据； 也可以利用 rpm -Va 来观察系统上面所有的已经被更动过的档案数据！

- 请找出 CentOS 内的 "Development Tools" 相关的数据，如果您尚未安装，请安装他！

  利用『 yum grouplist 』找出相关的套件群组，利用『 yum groupinfo "Develop Tools" 』找出该套件是否为您所需要， 最后用『 yum groupinstall "Develop Tools" 』安装即可！

- 你在进行 CentOS 安装时，选择的是预设安装，装完毕后才发现没有 gcc 这个编译器。 请问你如何安装 gcc ？

  可以使用『 yum search gcc 』找出所需要的套件，然后用『 yum install xxx 』来安装即可！

参考数据

- 台湾计算机危机处理小组(TWCERT)：http://www.cert.org.tw/
- Red Hat 的官方说明：http://www.redhat.com/apps/support/errata/
- APT 官方网站：http://apt.freshrpms.net/
- Rondo 的 APT 实做：http://163.19.59.1/~linux/student_samba/apt/apt_server.html
- APT Server：http://install.opennms.org/apt/
- 由 Red Hat 9 升级到 CentOS 4.1：http://www.owlriver.com/tips/centos-31-ex-rhl-9/
- twu2 兄提供的不同版本间升级：http://phorum.study-area.org/viewtopic.php?t=28648
- CentOS 官方网站提供的一些升级建议：
  http://www.centos.org/modules/newbb/viewtopic.php?topic_id=428&forum=6
  http://www.centos.org/modules/newbb/viewtopic.php?topic_id=382
- SuSE 的 YOU 自动升级机制须知：
  http://support.novell.com/linux/registration/
  http://support.novell.com/techcenter/articles/RegandUpdate_SLE10.html

---

路由器 Router 设定

最近更新日期：2006/08/30

我们在 网络基础 里面提到 路由 (route) 是一个重要的概念，他可以控制我们的资料封包的走向！ 此外，如果同一个网域里面有太多的计算机数量需要来广播的话，效能一定不会太好，所以才会有 Netmask 对吧！今天我们换个角度来想一想，如果说我的网域内真的有太多的计算机数量了， 那么将整个网域切割成较小的数个子网域 (Subnet) 会是一个比较好的作法，不过， 因为网域与网域之间的封包不可以直接互通数据，所以这个时候我们就需要使用 Router (路由器) 来帮忙封包的传送了！

路由

我们在网络基础里面谈到过路由的相关概念， 他最大的功能就是在帮我们规划网络封包的传递方式与方向。 至于路由的观察则可以使用 route 或者是 netstat 来查阅与设定。 好了，那么路由的形式有哪些？您又该如何确认路由是否正确呢？

路由表

如同前面网络基础谈到的，每一部主机都有自己的路由表， 也就是说，您必须要透过你自己的路由表来传递你主机的封包到下一个路由器上头。若传送出去后，该封包就得要透过下一个路由器的路由表来传送了，此时与你自己主机的路由表就没有关系啦！ 所以说，如果网络上面的某一部路由器设定错误，那.....封包的流向就会发生很大的问题。 我们就得要透过 traceroute 来尝试了解一下每个 router 的封包流向啰。

OK！那你自己主机的路由表到底有哪些部分呢？我们以底下这个路由表来说明：

```
[root@linux ~]# route -n
Kernel IP routing table
Destination   Gateway     Genmask        Flags Metric Ref  Use Iface
```

```
192.168.1.0    0.0.0.0      255.255.255.0  U     0      0        0 eth0 <== 1
127.0.0.0      0.0.0.0      255.0.0.0      U     0      0        0 lo   <== 2
0.0.0.0        192.168.1.2  0.0.0.0        UG    0      0        0 eth1 <== 3
```

首先，我们得知道在 Linux 系统下的路由表是由小网域排列到大网域， 例如上面的路由表当中，路由是由『 192.168.1.0/24 --> 127.0.0.0/8 --> 0.0.0.0/0 (预设路由) 』来排列的。 而当主机的网络封包需要传送时，就会查阅上述的三个路由规则来了解如何将该封包传送出去。

你会不会觉得奇怪，为什么会有这几个路由呢？其实路由表主要有这几种情况来设计的：

- 依据界面而存在的路由：
  例如 192.168.1.0/24 这个路由的存在是由于鸟哥的这部主机上面拥有 192.168.1.11 这个 IP 的关系！ 也就是说，你主机上面有几个网络接口的存在时，该网络接口就会存在一个路由才对。所以说，万一你的主机有两个网络接口时，例如 192.168.1.11, 192.168.2.11 时，那路由至少就会有：

```
[root@linux ~]# route -n
Kernel IP routing table
Destination   Gateway      Genmask        Flags Metric Ref   Use Iface
192.168.1.0   0.0.0.0      255.255.255.0  U     0      0       0 eth0
192.168.2.0   0.0.0.0      255.255.255.0  U     0      0       0 eth1
.....以下省略.....
```

- 手动或预设路由(default route)：
  你可以使用 route 这个指令手动的给予额外的路由设定，例如那个预设路由 (0.0.0.0/0) 就是额外的路由。 使用 route 这个指令时，最重要的一个概念是：『你所规划的路由必须要是你的装置 (如 eth0) 或 IP 可以直接沟通 (broadcast) 的情况』才行。举例来说，以上述的环境来看， 我的环境里面仅有 192.168.1.11 及 192.168.2.11 ，那我如果想要连接到 192.168.100.254 这个路由器时， 下达：

```
[root@linux ~]# route add -net 192.168.100.0  \
> netmask 255.255.255.0 gw 192.168.100.254
SIOCADDRT: Network is unreachable
```

  看吧！系统就会响应没有办法连接到该网域，因为我们的网络接口与 192.168.100.0/24 根本就没有关系嘛！ 那如果 192.168.100.254 真的是在我们的实体网络连接上，那其实你应该是这样做：

```
[root@linux ~]# route add -net 192.168.100.0  \
> netmask 255.255.255.0 dev eth0
```

```
[root@linux ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref Use Iface
192.168.100.0   0.0.0.0         255.255.255.0   U     0      0     0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0     0 eth0
0.0.0.0         192.168.1.2     0.0.0.0         UG    0      0     0 eth1
```

这样你的主机就会直接用 eth0 这个装置去尝试连接 192.168.100.254 了！ 另外，上面路由输出的重点其实是那个『Flags 的 G 』了！因为那个 G 代表使用外部的装置作为 Gateway 的意思！而那个 Gateway (192.168.1.2) 必须要在我们的已存在的路由环境中。 这可是很重要的概念喔！^_^

- 动态路由：
  除了上面这两种可以直接使用指令的方法来增加路由规则之外，还有一种透过路由器与路由器之间的协商以达成动态路由的环境， 不过，那就需要额外的软件支持了，例如： zebra (http://www.zebra.org/) 或 CentOS 上面的 Quagga (http://www.quagga.net/) 这几个软件了！

事实上，在 Linux 的路由规则都是透过核心来达成的， 所以这些路由表的规则都是在核心功能内啊！也就是在内存当中喔！ ^_^

IP Alias 的测试用途

我们在 Linux 常用指令里面谈过 eth0:0 这个装置吧？ 这个装置可以在原本的 eth0 上面模拟出一个虚拟接口出来，以让我们原本的网络卡具有多个 IP ， 具有多个 IP 的功能就被称为 IP Alias 了。而这个 eth0:0 的装置可以透过 ifconfig 或 ip 这两个指令来达成， 关于这两个指令的用途请翻回去之前的章节阅读，这里不再浪费篇幅啊！

那你或许会问啊：『这个 IP Alias 有啥用途啊？』好问题！这个 IP Alias 最大的用途就是可以让你用来『应急』！ 怎么说呢？我们就来聊一聊他的几个常见的用途好了：

- 测试用：
  怎么说用来测试呢？举例来说，现在使用 IP 分享器的朋友很多吧， 那 IP 分享器通常使用 WWW 接口来提供设定。那这个 IP 分享器通常会给予一个私有 IP 亦即是 192.168.0.1 来让使用者开启 WWW 接口的浏览。问题来了，那你要如何连接上这部 IP 分享器呢？ 嘿嘿！在不更动既有的网络环境下，你可以直接利用：

```
[root@linux ~]# ifconfig [device] [ IP ] netmask [netmask ip] [up|down]
[root@linux ~]# ifconfig eth0:0 192.168.0.100 netmask 255.255.255.0 up
```

来建立一个虚拟的网络接口，这样就可以立刻连接上 IP 分享器了。

- 在一个实体网域中含有多个 IP 网域：
  另外，如果像是在补习班或者是学校单位的话， 由于原本的主机网络设定最好不要随便修改，那如果要让同学们大家互通所有的计算机信息时， 就可以让每个同学都透过 IP Alias 来设定同一网域的 IP ， 如此大家就可以在同一个网段内进行各项网络服务的测试了，很不错吧！

- 既有设备无法提供更多实体网卡时：
  如果你的这部主机需要连接多个网域，但该设备却无法提供安装更多的网卡时， 你只好勉为其难的使用 IP Alias 来提供不同网段的联机服务了！

不过，你需要知道的是：所有的 IP Alias 都是由实体网卡仿真来的， 所以当要启动 eth0:0 时，eth0 必须要先被启动才行。而当 eth0 被关闭后， 所以 eth0:n 的模拟网卡将同时也被关闭。这得先要了解才行，否则常常会搞错启动的装置啊！ 在路由规则的设定当中，常常需要进行一些测试，那这个 IP Alias 就派的上用场了。 尤其是学校单位的练习环境当中！ ^_^！

基本上，除非有特殊需求，否则建议你要有多个 IP 时，最好在不同的网卡上面达成， 如果你真的要使用 IP Alias 时，那么如何在开机的时候就启动 IP alias 呢？有两个简单的方法可以使用：

- 透过 /etc/rc.d/rc.local：
  将『ifconfig eth0:n ....』的指令写入 /etc/rc.d/rc.local 当中，这样开机的时候就能够启动这个虚拟接口， 不过这方法有个弱点，就是当使用类似『 /etc/init.d/network restart 』的指令时，该接口可能就会被取消。

- 透过 /etc/sysconfig/network-scripts/ifcfg-eth0:0：
  举例来说，你可以透过底下这个方法来建立一个虚拟装置的设定档案：

```
[root@linux ~]# cd /etc/sysconfig/network-scripts
[root@linux network-scripts]# vi ifcfg-eth0:0
DEVICE=eth0:0            <==相当重要！一定要与文件名相同的装置代号！
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.0.100
NETMASK=255.255.255.0
NETWORK=192.168.0.0
BROADCAST=192.168.0.255

[root@linux network-scripts]# ifup eth0:0
[root@linux network-scripts]# ifdown eth0:0
```

  关于装置的设定档案内的更多参数说明，请参考连上 Internet 一文的相关说明，在此不再叙述！使用这个方法有个好处，就是当你使用『 /etc/init.d/network restart 』时， 系统依旧会使用你的 ifcfg-eth0:0 档案内的设定值来启动你的虚拟网卡喔！另外， 不论 ifcfg-eth0:0 内的 ONBOOT 设定值为何，只要 ifcfg-eth0 这个实体网卡的设定文件中， ONBOOT 为 yes 时，开机就会将全部的 eth0:n 都启动。

透过这两个简单的方法，你就可以在开机的时候启动你的虚拟接口而取得多个 IP 在同一张网卡上了。 不过依旧要注意的是，如果你的 eth0 是使用 DHCP 来取得 IP 参数的话，那么由于 ifup 及 /etc/init.d/network 这两个 script 内程序代码撰写的方式，将会导致 ifcfg-eth0:0 这个设定档不会被使用到喔！ 所以当你使用 DHCP 来取得 eth0 的 IP 时，你只好使用手动方式（用 ifconfig）来设定你的 IP Alias 了。

重复路由的问题

很多朋友可能都有一个可爱的想法，那就是：『我可不可以利用两张网卡， 利用两个相同网域的 IP 来增加我这部主机的网络流量』？事实上这是一个可行的方案， 不过必须要透过许多的设定来达成，若您有需求的话，可以参考网中人大哥写的这一篇：

- 频宽负载平衡 (http://www.study-area.org/tips/multipath.htm)

如果只是单纯的以为设定好两张网卡的 IP 在同一个网域而已，那您可就大错特错了～ 为什么呢？还记得我们在路由表规则里面提过网络封包的传递主要是依据主机内的路由表规则吧！ 那如果你有两张网络卡时，假设：

- eth0 : 192.168.0.100
- eth1 : 192.168.0.200

那你的路由规则会是如何呢？理论上会变成这样：

```
[root@linux ~]# route -n
Kernel IP routing table
Destination     Gateway     Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0     255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0     255.255.255.0   U     0      0        0 eth0
```

也就是说，当要传送到 192.168.0.0/24 的网域时，都只会透过第一条规则，也就是透过 eth1 来传出去，而不管是由 eth0 还是由 eth1 进来的网络封包都会透过 eth1 来回传，这可能会造成一些问题， 尤其是一些防火墙的规则方面，很可能会发生一些严重的错误， 如此一来，根本没有办法达成负载平衡，也不会有增加网络流量的效果！ 更惨的是，还可能发生封包传递错误的情况呐！所以说，同一部主机上面设定相同网域的 IP 时， 得要特别留意你的路由规则，一般来说， 不应该设定同一的网段的不同 IP 在同一部主机上面。 例如上面的案例就是一个不好的示范啊！

路由器架设

在同一的局域网络里面可以透过广播 (broadcast) 了解到 MAC 与 IP 的解析，然后透过 MAC 对 MAC 来传送数据封包， 在不同的网域里头就得要透过路由器的帮忙。那么什么是路由器？他的主要功能是什么？底下我们就来聊一聊！

什么是路由器

既然主机想要将数据传送到不同的网域时得透过路由器的帮忙，所以啦，路由器的主要功能就是：『转递网络封包』啰！ 也就是说，路由器会分析来源端封包的 IP 表头，找出目标的 IP 后，透过路由器本身的路由表 (routing table) 将这个封包向下一个目标 (next hop) 传送。这就是路由器的功能。

那么路由器的功能可以如何达成呢？目前有两种方法可以达成：

- 硬件功能：例如 Cisco, IBM, 3Com 等公司都有生产硬件路由器，这些路由器内有嵌入式的操作系统， 可以负责不同网域间的封包转译与转递等功能；
- 软件功能：例如 Linux 这个操作系统的核心就有提供封包转递的能力。

高阶的路由器可以连结不同的硬设备，并且可以转译很多不同的封包格式，通常....价格也不便宜啊！ 在这个章节里面，我们并没有要探讨这么高阶的咚咚，仅讨论在以太网络里头最简单的路由器功能： 连接两个不同的网域。嘿嘿！这个功能 Linux 就可以达成了！ 就如同路由表是由 Linux 的核心功能所提供的，这个转递封包的能力也是 Linux 核心所提供， 那如何启动这个封包转递呢？很简单啊，只要这样做即可：

```
[root@linux ~]# echo "1" > /proc/sys/net/ipv4/ip_forward
```

上面这个动作就在打开 Linux 核心的封包转递能力。你可以将上述的指令写入 /etc/rc.d/rc.local 当中， 以使 Linux 可以在开机的时候就启动封包转递的功能， 也可以透过修改 /etc/sysctl.conf 来达成开机启动封包转递：

```
[root@linux ~]# vi /etc/sysctl.conf
# 将底下这个设定值修改正确即可！
net.ipv4.ip_forward = 1

[root@linux ~]# sysctl -p  <==立刻让该设定生效
```

sysctl 这个指令是在核心工作时用来直接修改核心参数的一个指令，更多的功能可以参考 man sysctl 查询。 不要怀疑！只要这个动作，你的 Linux 就具有最简单的路由器功能了。 而由于 Linux 路由器的路由表设定方法的不同，通常路由器规划其路由的方式就有两种：

- 静态路由：直接以类似 route 这个指令来直接设定路由表到核心功能当中，设定值只要与网域环境相符即可。 不过，当你的网域有变化时，路由器就得要重新设定；

- 动态路由：透过类似 zebra 软件的功能，这些软件可以安装在 Linux 路由器上， 而这些软件可以动态的侦测网域的变化，而直接修改 Linux 核心的路由表信息， 你无须手动以 route 来修改你的路由表信息喔！

了解了路由器之后，接下来你可能需要了解到什么是 NAT (Network Address Translation, 网络地址转译) 主机， NAT 是啥？其实 IP 分享器就是最简单的 NAT 主机啦！嘿嘿，了解了吗？没错， NAT 可以达成 IP 分享的功能， 而 NAT 本身就是一个路由器，但 NAT 比路由器多了一个『 IP 转换』的功能。怎么说呢？

- 一般来说，路由器会有两个网络接口，透过路由器本身的 IP 转递功能让两个网域可以互相沟通网络封包。 那如果两个接口一边是公共 IP (public IP) 但一边是私有 IP (private IP) 呢？ 由于私有 IP 不能直接与公共 IP 沟通其路由信息，此时就得要额外的『 IP 转译』功能了；

- Linux 的 NAT 主机可以透过修改封包的 IP 表头数据之来源或目标 IP ，让来自私有 IP 的封包可以转成 NAT 主机的公共 IP ，就可以连上 Internet ！

所以说， 当路由器两端的网域分别是 Public 与 Private IP 时，才需要 NAT 的功能！ NAT 功能我们会在防火墙时谈及，这个章节仅谈论一下路由器而已啊！ ^_^

何时需要路由器？

一般来说，计算机数量小于数十部的小型企业是无须路由器的，只需要利用 hub/switch 串接各部计算机，然后透过单一线路连接到 Internet 上即可。不过，如果是超过数百部计算机的大型企业环境， 由于他们的环境通常需要考虑如下的状况，因此才需要路由器的架设：

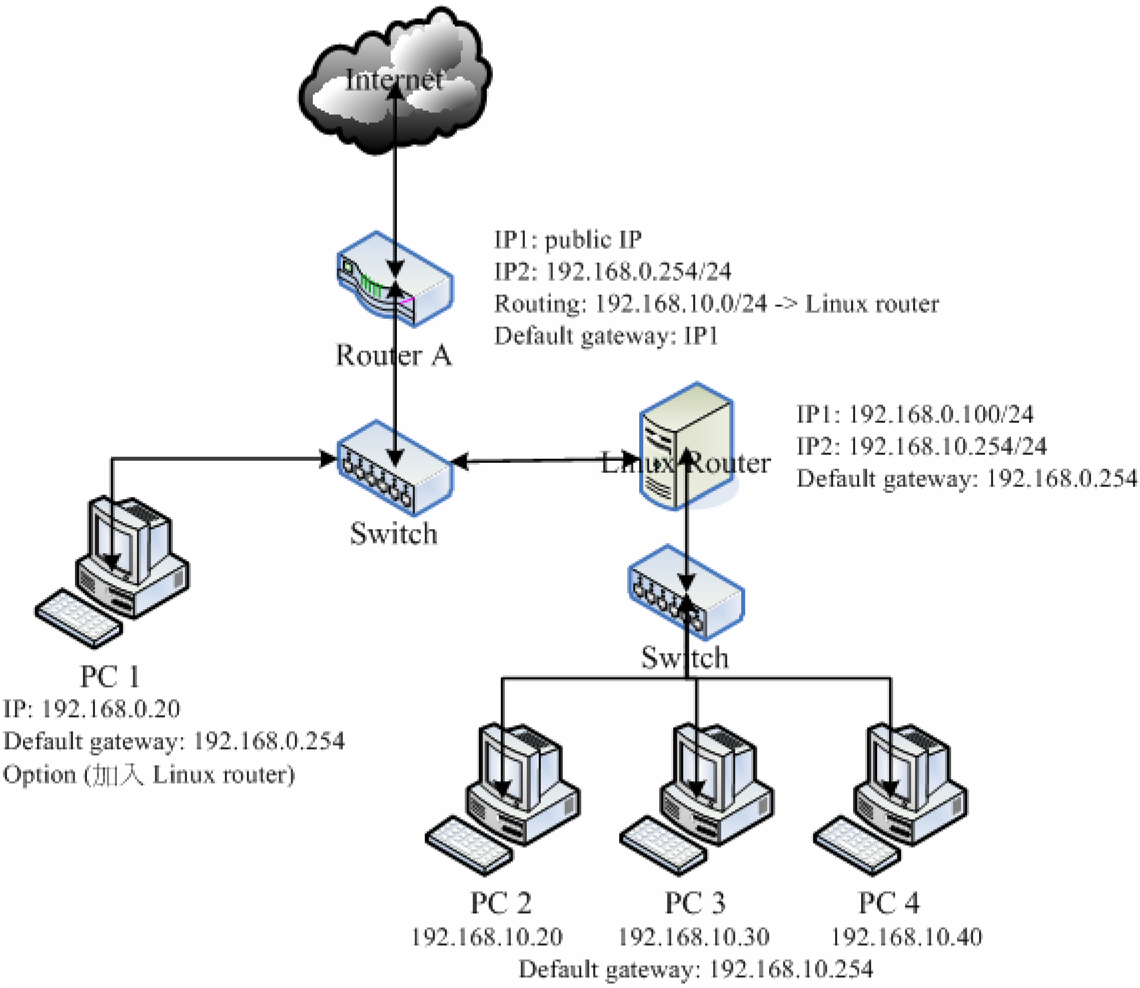
- 实体线路之布线及效能的考虑：
  在一栋大楼的不同楼层要串接所有的计算机可能有点难度，那可以透过每个楼层架设一部路由器，并将每个楼层路由器相连接，就能够简单的管理各楼层的网络； 此外，如果各楼层不想架设路由器，而是直接以网络线串接各楼层的 hub/switch 时， 那由于同一网域的数据是透过广播来传递的，那当整个大楼的某一部计算机在广播时， 所有的计算机将会予以响应，哇！会造成大楼内网络效能的问题；所以架设路由器将实体线路分隔， 就有助于这方面的网络效能；

- 部门独立与保护数据的考虑：
  在阅读过网络基础章节后，您就会晓得， 只要实体线路是连接在一起的，那么当数据透过广播时，你就可以透过类似 tcpdump 的指令来监听封包数据， 并且予以窃取～所以，如果你的部门之间的数据可能需要独立， 或者是某些重要的数据必须要在公司内部也予以保护时，可以将那些重要的计算机放到一个独立的实体网域， 并额外加设防火墙、路由器等连接上公司内部的网域。

路由器就只是一个设备，要如何使用端看您的网络环境的规划！上面仅是举出一些应用案例。 底下我们先就架设一个静态路由的路由器来玩一玩吧！

静态路由之路由器

假设在贵公司的网络环境当中，除了一般职员的工作用计算机是直接连接到对外的路由器来连结 Internet， 在内部其实还有一个部门需要较安全的独立环境，因此这部份的网络规划可能是这样的情况：



图一、静态路由之路由器架构示意图

以上图的架构来说，这家公司主要有两个 C class 的网域，分别是 192.168.0.0/24 及 192.168.10.0/24 ， 其中 192.168.0.0/24 是用来做为一般员工连接因特网用的，至于 192.168.10.0/24 则是给特殊的部门用的。PC1 代表的是一般员工的计算机， PC2 及 PC3, PC4 则是特殊部门的工作用计算机， Linux Router 则是这个特殊部门用来连接到公司内部网域的路由器。在这样的架构下， 该特殊部门的封包就能够具有基础的保护了。

由图一你也不难发现，只要是具有路由器功能的设备 (Router A, Linux Router) 都会具有两个以上的接口， 分别用来沟通不同的网域，同时该路由器也都会具有一个预设路由啊！ ^_^！ 另外，你还可以加上一些防火墙的软件在 Linux Router 上， 以保护 PC2~PC4 的内部计算机呢！

在 Router A 的部分，由于他具有 Public 与 Private IP ，所以这部 Router 必须要具有 NAT 的功能，这个未来我们再介绍。今日的重点就在于 Linux Router 那个玩意儿！在该主机下， 最好配备两张网卡，一张给 192.168.0.100，另一张给 192.168.10.254 这个 IP。 这部 Linux Router 的设定简单的要命喔！你可以这样做的：

- Linux Router

在这部主机内需要有两张网卡，鸟哥在这里将他定义为：

- eth0: 192.168.10.254
- eth1: 192.168.0.100

那如何设定呢？这样做就对了：

```
1. 先处理 eth0
[root@linux ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.10.255
IPADDR=192.168.10.254
NETMASK=255.255.255.0
NETWORK=192.168.10.0
ONBOOT=yes

2. 再先处理 eth1
[root@linux ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
BROADCAST=192.168.0.255
IPADDR=192.168.0.100
NETMASK=255.255.255.0
NETWORK=192.168.0.0
GATEWAY=192.168.0.254    <==这个设定值很重要喔！
ONBOOT=yes

3. 启动 IP 转递
[root@linux ~]# echo "1" > /proc/sys/net/ipv4/ip_forward
# 上述指令如果没有问题，将他加入 /etc/rc.d/rc.local 当中去！

4. 重新启动网络，并且观察路由
[root@linux ~]# /etc/init.d/network restart
[root@linux ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref Use Iface
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0     0 eth1
192.168.10.0    0.0.0.0         255.255.255.0   U     0      0     0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0     0 lo
0.0.0.0         192.168.0.254   0.0.0.0         UG    0      0     0 eth1
```

有够简单吧！这样你的 Linux Router 就 OK 了呐！接下来则是 PC2 来作为范例。

- 受保护的网域，以 PC2 为例：

不论你的 PC2 是哪一种操作系统，你的环境都应该是这样的：

- IP: 192.168.10.20
- netmask: 255.255.255.0
- network: 192.168.10.0
- broadcast: 192.168.10.255
- gateway: 192.168.10.254

以 Linux 操作系统为例，并且 PC2 仅有 eth0 一张网卡时，他的设定是这样的：

```
[root@linux ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.10.255
IPADDR=192.168.10.20
NETMASK=255.255.255.0
NETWORK=192.168.10.0
GATEWAY=192.168.10.254   <==这个设定最重要啦！
ONBOOT=yes

[root@linux ~]# /etc/init.d/network restart
[root@linux ~]# route -n
Kernel IP routing table
Destination   Gateway        Genmask        Flags Metric Ref   Use Iface
192.168.10.0  0.0.0.0        255.255.255.0  U     0      0       0 eth0
127.0.0.0     0.0.0.0        255.0.0.0      U     0      0       0 lo
0.0.0.0       192.168.10.254 0.0.0.0        UG    0      0       0 eth0
```

- Router A 的新增路由规则：

在这样的架构下，您的 PC2 已经可以连接上 Internet 了！只不过，当封包由 Internet 传回来时， 由于 Router A 并没有连接到 192.168.10.0/24 网域的路由规则，所以该封包『会遗失』喔！那就麻烦了～所以，你的 Router A 必须要额外增加一条规则，这条规则是『将目标为 192.168.10.0/24 的封包传送给 192.168.0.100 去处理』，假设 Router A 为 Linux 系统时，那他应该要这样：

```
[root@linux ~]# route add -net 192.168.10.0 netmask 255.255.255.0 \
>  gw 192.168.0.100

[root@linux ~]# route -n
Destination   Gateway        Genmask        Flags Metric Ref   Use Iface
192.168.10.0  192.168.0.100  255.255.255.0  UG    0      0       0 eth0
# 你应该会发现上头这一条规则存在才行！
```

如此一来，你的 Router A 及 Linux Router 之间就可以沟通，并且可以传递 192.168.10.0/24 的网域呐！

- PC1 直接与 PC2 的沟通方式：

在图一当中，你会发现那个 PC1 必须要将预设路由设定为 192.168.0.254 ， 所以当 PC1 要与 PC2 沟通时，封包将由：

PC1 --> Router A --> Linux Router --> PC2

不过，在图一当中我们知道其实 PC1 与 PC2 同时接到 Linux Router 上嘛！ 所以 PC1 其实可以直接加入一条路由规则，规定当 192.168.10.0/24 的封包目标时， 他可以直接传到 Linux Router 上即可，那可以这样做：

```
[root@linux ~]# route add -net 192.168.10.0 netmask 255.255.255.0 \
>  gw 192.168.0.100
```

最后只要 PC2 使用 ping 可以连到 PC1，同样的，PC1 也可以 ping 到 PC2 的话，就表示你的设定是 OK 的啦！嘿嘿！搞定！而透过这样的设定方式，您也可以发现到一件事， 那就是：『路由是双向的，你必须要了解出去的路由与回来时的规则』。 举例来说，在预设的情况下 (Router A 与 PC1 都没有额外的路由设定时)，其实封包是可以由 PC2 联机到 PC1 的，但是 PC1 却没有相关的路由可以响应到 PC2 ～所以上头才会要您在 Router A 或者是 PC1 上面设定额外的路由规则啊！这样说，瞭了吧？ ^_^

所以说，用 Linux 作一个静态路由的 Router 很简单吧！以上面的案例来说，你在 Linux Router 上面几乎没有作什么额外的工作，只要将网络 IP 与网络接口对应好启动，然后加上 IP Forward 的功能， 让你的 Linux 核心支持封包转递，然后其它的工作咱们的 Linux kernel 就主动帮你搞定了！ 真是好简单！ ^_^

不过这里必须要提醒的是，如果你的 Linux Router 有设定防火墙的话， 而且还有设定类似 NAT 主机的 IP 伪装技术，那可得特别留意，因为还可能会造成路由误判的问题～ 上述的 Linux Router 当中『并没有使用到任何 NAT 的功能』喔！特别给他留意到！

动态路由之路由器设定：zebra

如前所述，系统管理员可以利用 route 这个指令手动的将路由规则加入核心当中，这个方式称为静态路由。动态路由同样是将路由规则加入核心当中，只是这个加入的动作交由软件服务 (daemon) 自动来执行， 在 Linux 上面常见的路由服务就是 zebra 这个套件所提供的。

动态路由通常是用在路由器与路由器之间的沟通，所以要让您的路由器具有动态路由的功能， 你必须要了解到对方路由器上面所提供的动态路由协议才行，这样两部路由器才能够透过该协议来沟通彼此的路由规则。 目前常见的动态路由协议有：RIPv1, RIPv2, OSPF, BGP 等等， zebra 都有支持这些路由协议喔！

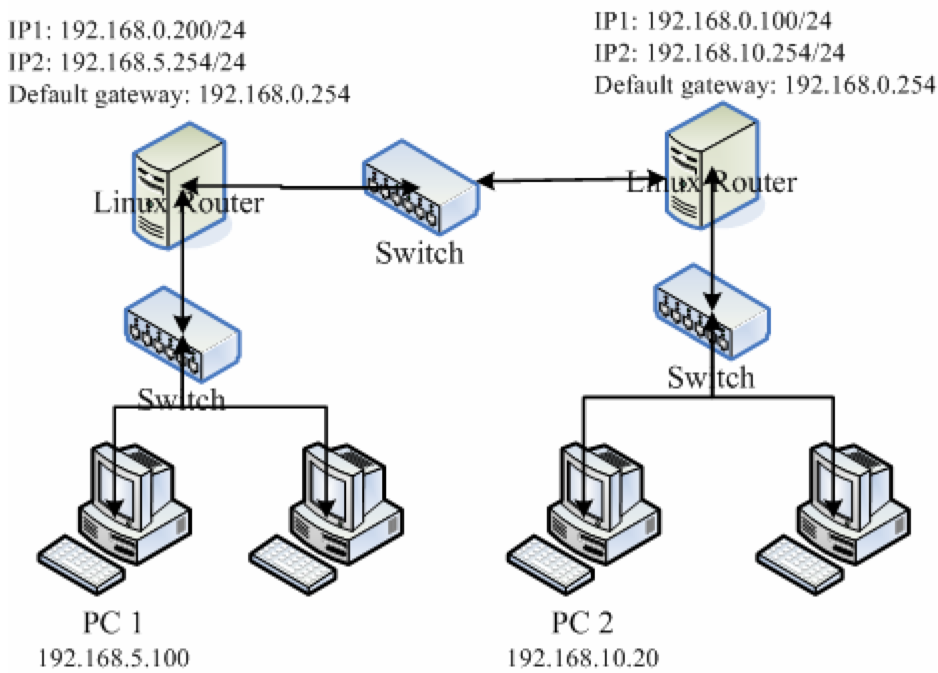
在 CentOS 上头，我们使用 quagga 这个套件来提供 zebra ，因为 quagga 是更新 zebra 而来的， 事实上，你可以说， quagga 就是 zebra 啦！赶紧安装他先：

```
[root@linux ~]# yum install quagga
[root@linux ~]# ls -l /etc/quagga
-rw-r--r--  1 root    root      410 Jun  2 02:38 ripd.conf.sample
-rw-r-----  1 quagga quagga      30 Aug 29 10:50 zebra.conf
-rw-r--r--  1 root    root      373 Jun  2 02:38 zebra.conf.sample
.....其它省略.....
```

这个套件所提供的各项动态路由协议都放置到 /etc/quagga/ 目录内，底下我们以较为简单的 RIPv2 协议来处理动态路由， 不过你得要注意的是，不论你要启动什么动态路由协议，那个 zebra 都必须要先启动才行！ 这是因为：

- zebra 这个 daemon 的功能在更新核心的路由规则；
- RIP 这个 daemon 则是在向附近的其它 Router 沟通协调路由规则的传送与否。

而各个路由服务的设定档都必须要以 /etc/quagga/*.conf 的档名来储存才行，如上表我们可以发现 zebra 这个服务是有设定好了，不过 ripd 的档名却不是 .conf 结尾。所以我们必须要额外作些设定才行。而假设我们的网络连结如下图二所示：



图二、动态路由的简易图标

这两部 Linux Router 分别负责不同的网域，且可以透过 192.168.0.0/24 这个网域来沟通。 在没有设定额外路由规则的情况下，那个 PC1 与 PC2 是无法沟通的！另外， zebra 必须要同时安装在两部 Linux Router 上头才行， 而且我们只要设定好这两部主机的网络接口 (eth0, eth1) 后，不需要手动输入额外的路由设定喔！可以透过 RIP 这个路由协议来搞定的！

- 设定 zebra

我们先设定图二右手边那一部 Linux Router，关于 zebra.conf 你可以这样设定的：

```
1. 先设定 zebra 并且启动 zebra
[root@linux ~]# vi /etc/quagga/zebra.conf
hostname linux.router1  <==给予这个路由器一个主机名称，随便取！
password linux1         <==给予一个密码！
enable password iinux1  <==将这个密码生效！
log file zebra.log      <==将所有 zebra 产生的信息存到登录文件中
```

```
[root@linux ~]# /etc/init.d/zebra start
[root@linux ~]# netstat -tunlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address   State   PID/Program name
tcp        0      0 127.0.0.1:2601  0.0.0.0:*         LISTEN  6422/zebra
```

仔细看，由于 zebra 这个服务的任务主要是在修改 Linux 系统核心内的路由， 所以他仅监听本机接口而已，并不会监听外部的接口才对！另外，在 zebra.conf 这个档案当中， 我们所设定的那个密码是有作用的喔！可以让我们登入 zebra 这套软件呢！ 好了，我们来查一查这个 2601 的 port 是否正确的启动的呢？

```
[root@linux ~]# telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.98.3).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password: <==在这里输入刚刚你设定的密码啊！
linux.router1> <==在这边输入『 ? 』就能够知道有多少指令可使用
  echo      Echo a message back to the vty
  enable    Turn on privileged mode command
  exit      Exit current mode and down to previous mode
  help      Description of the interactive help system
  list      Print command list
  quit      Exit current mode and down to previous mode
  show      Show running system information
  terminal  Set terminal line parameters
  who       Display who is on vty
linux.router1> list
  echo .MESSAGE
  enable
  exit
  help
  list
  quit
  show debugging zebra
  show history
  show interface [IFNAME]
  show ip forwarding
  show ip route
```

```
....其它省略....
linux.router1> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

K>* 0.0.0.0/0 via 192.168.0.254, eth0
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.0.0/24 is directly connected, eth0
C>* 192.168.10.0/24 is directly connected, eth1
linux.router1> exit
Connection closed by foreign host.
```

仔细看到，我们登入这个 zebra 的软件之后，可以输入『help』或问号『?』， zebra 就会显示出你能够执行的指令有哪些，比较常用的当然是查询路由规则啰！ 以『 show ip route 』来查阅，结果可以发现目前的接口与预设路由都被显示出来了， 显示的结果当中，K 代表以 router 这个指令直接加入核心的路由规则，C 则代表你的网络接口相关的路由规则。

事实上，如果你还想要增加额外的静态路由的话，也可以透过 zebra 而不必使用 route 指令呢！ 例如想要增加 10.0.0.0/24 给 eth0 来处理的话，可以这样做：

```
[root@linux ~]# vi /etc/quagga/zebra.conf
# 新增底下这一行喔！
ip route 10.0.0.0/24 eth0

[root@linux ~]# /etc/init.d/zebra restart
[root@linux ~]# telnet localhost 2601
User Access Verification

Password: <==这里输入密码
linux.router1> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

K>* 0.0.0.0/0 via 192.168.0.254, eth0
S>* 10.0.0.0/24 [1/0] is directly connected, eth0
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.0.0/24 is directly connected, eth0
C>* 192.168.10.0/24 is directly connected, eth1
```

嘿嘿！立刻就会多出一笔路由的规则，而且最右边会显示 S，亦即是静态路由 (Static route) 的意思。 如此一来，我们系统管理员可就轻松多了！ 设定完 zebra 之后，接下来我们可以开始看看 ripd 这个服务啰！

- 设定 ripd 服务

ripd 这个服务可以在两部 Router 之间进行路由规则的交换与沟通， 当然啦，如果你的环境里面有类似 Cisco 或者是其它有提供 RIP 协议的路由器的话， 那么你当然也是可以透过这个 RIP 让您的 Linux Router 与其它硬件路由器互相沟通的呐！ 闲话少说，来设定 ripd 吧！

```
[root@linux ~]# vi /etc/quagga/ripd.conf
hostname linux.router1  <==这里是设定 Router 的主机名称而已
password linux1         <==设定好你自己的密码喔！
router rip              <==启动 Router 的 rip 功能
network 192.168.0.0/24  <==针对这个网域来进行监听的动作！
network eth0            <==针对这个接口来进行监听的动作
network 192.168.10.0/24 <==针对这个网域来进行监听的动作！
network eth1            <==针对这个接口来进行监听的动作
version 2               <==启动的是 RIPv2 的服务
log stdout              <==直接在屏幕输出标准输出的数据

[root@linux ~]# /etc/init.d/ripd start

[root@linux ~]# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address State  PID/Program name
tcp        0      0 0.0.0.0:2602   0.0.0.0:*       LISTEN 21373/ripd
```

基本上，这样就设定完成一部路由器的 RIP 动态路由协议了！在上头 ripd.conf 的设定当中， 他会主动以 eth0 及 192.168.0.0/24 这个网域的功能来进行搜索，如此一来，未来你进行任何路由规则的变动，或者是整个网域的主机 IP 进行更动，你将不需要重新到每部 Router 上更动！ 因为这些路由器会自动的更新他们自己的规则喔！嘿嘿！接下来，同样的动作请你到 图二左边那部 Linux Router 上面设定一下！ 因为整个设定的流程都一样，所以这里鸟哥就省略啦！

- 检查 RIP 协议的沟通结果

在两部 Linux Router 都设定妥当之后，你可以登入 zebra 去看这两部主机的路由更新结果喔！ 举例来说，鸟哥登入图二右边那部 Linux Router 后，并且登入 zebra ， 观察路由会是这样的情况：

```
[root@linux ~]# telnet localhost 2601
User Access Verification

Password: <==不要忘记了密码啊！
linux.router1> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       B - BGP, > - selected route, * - FIB route

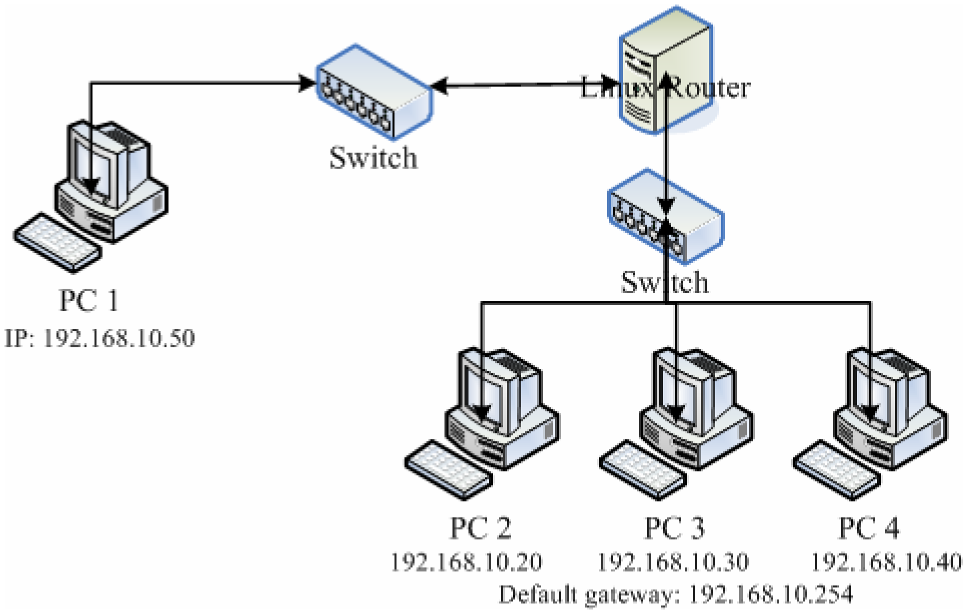
K>* 0.0.0.0/0 via 192.168.0.254, eth0
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.0.0/24 is directly connected, eth0
R>* 192.168.5.0/24 [120/2] via 192.168.0.200, eth0, 00:06:48
C>* 192.168.10.0/24 is directly connected, eth1
```

如果你有看到上述的字体，嘿嘿！那就是成功啦！那个最左边的 R 代表的是透过 RIP 通讯协议所设定的路由规则啦！ 如此一来，咱们的路由器设定就搞定啰～

透过这个 zebra 以及 RIPv2 的路由协议的辅助，我们可以轻松的就将路由规则分享到附近区网的其它路由器上头， 比起单纯使用 route 去修改 Linux 的核心路由表，这个动作当然要快速很多！ 不过，如果是很小型的网络环境，那么不要使用这个 zebra 啊！因为有点多此一举的感觉。 如果您的企业环境真的有够大，那么玩一玩这个 zebra 配合一些动态路由协议，嘿嘿！也是可行的啦！

ARP Proxy 让路由器两端在同一网域

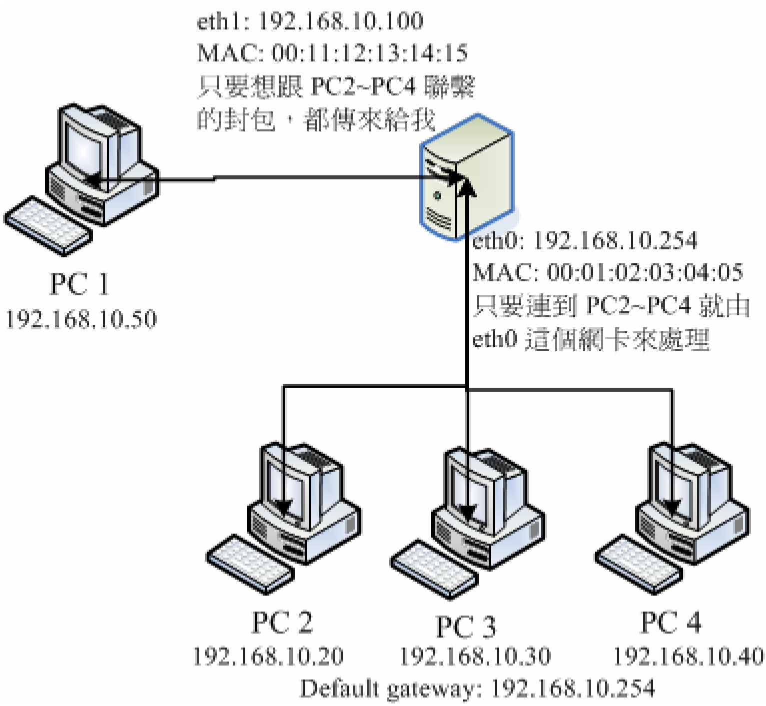
如果你一开始设计的网络环境就是同一个 C class 的网域，例如 192.168.10.0/24 ， 后来因为某些因素必须要将某些主机搬到比较内部的环境中，例如图一的 PC2 ~ PC4 。 然后又因为某些因素，所以你不能变更 PC2 ~ PC4 的 IP ，也就是说，有点像底下这样的图示：



图三、路由器两边是同一网域的特殊状况

初次见面～看到眼睛快要掉下来哩！怎么会两边的主机都在同一个网域内？而且还被规定不能够更改原先的 IP 设定， .....真是一个头两个大啊～如此一来，在 Linux Router 两边要如何制作路由啊？好问题！真是好问题～ 因为 OSI 第三层网络层的路由是一条一条去设定比对的，所以如果两块网卡上面都是同一个网域的 IP 时， 就会发生错误。那如何处理啊？

既然 OSI 第三层无法解决，那么可否以第二层的数据连接层来处理？看倌应该还记得 OSI 第二层最重要的就是那个 ARP 协议， 他可以用来进行 IP 与 MAC 的对应。那么由图三我们知道 PC1 要与 PC2 等主机沟通时，都需要透过 Linux Router ， 那有没有办法透过 ARP 告知整个网域内的计算机，要传送到 PC2~PC4 的封包都需要经过 Linux Router 呢？ 呵呵！好想法。你可以这样想：



图四、路由器两边是同一网域的特殊状况

也就是说：

1. 当 Linux Router 的 eth1 那个网域主机想要连接到 PC2~PC4 的主机时，由 Linux Router 负责接收；
2. 当 Linux Router 要传送数据到 PC2~PC4 时，务必要由 eth0 来传送；
3. 当 Linux Router 要传送的数据为 192.168.10.0/24 ，但并非 PC2~PC4 时，需由 eth1 传送；
4. 当 Linux Router 的 eth0 那个网域主机想要连接到 PC1 时，由 Lniux Router 负责接收。

要达到 (1) 与 (4) 的要求并不难，我们可以透过 ARP Proxy 这玩意儿，啥是 ARP Proxy 呢？就是在 Linux Router 上面预先规定『将 192.168.10.20, 192.168.10.30, 192.168.10.40 这三个 IP 的 MAC 都对应到 Linux Router 上！』由于是局域网络内，因此都是透过广播的方式达到 ARP 协议所需要的 IP 与 MAC 的对应，所以啦，每一部在 eth1 那端的主机都会『误判』那三个 IP 是 Linux Router 所拥有， 这样就能够让封包传给 Linux Router 啦！

再接下来，咱们的 Linux Router 必须要额外指定路由，设定：

- 若目标是 PC2 ~ PC4 时，该路由必须要由 eth0 发送出去才行，
- 若目标不为 PC2 ~ PC4 ，且目标在 192.168.10.0/24 的网域时，需由 eth0 发送出去才行。

也就是说，你必须要指定路由规则当中，那个 PC2~PC4 具有优先选择权，然后其它的同网域封包才由 eth1 来传送。 这样就能够达成我们所想要的结局啦！^_^！看样子似乎很难，其实设定方面还挺简单的，你可以透过 arp 以及 route 这两个指令来达成喔！

```
1. 先设定 ARP Proxy ，告知 eth1 所在网域 IP 与 MAC 的对应
[root@linux ~]# arp -i eth1 -s 192.168.10.20 00:11:12:13:14:15 pub
[root@linux ~]# arp -i eth1 -s 192.168.10.30 00:11:12:13:14:15 pub
```

```
[root@linux ~]# arp -i eth1 -s 192.168.10.40 00:11:12:13:14:15 pub
# 看图四的说明，我这里假设 eth1 的 MAC 是 00:11:12:13:14:15 啦！
[root@linux ~]# arp -i eth0 -s 192.168.10.50 00:01:02:03:04:05 pub
# 看图四的说明，我这里假设 eth0 的 MAC 是 00:01:02:03:04:05 啦！

[root@linux ~]# arp -n
Address             HWtype  HWaddress       Flags Mask        Iface
192.168.10.20       *       *               MP                eth1
192.168.10.30       *       *               MP                eth1
192.168.10.40       *       *               MP                eth1
192.168.10.50       *       *               MP                eth0
# 瞧！有三个 IP 都变成属于俺的 eth1 的啦！然后一个属于 eth0

2. 开始处理路由，需要清除掉 eth0 的路由，并且增加 PC2~PC4 的单机路由
[root@linux ~]# route del -net 192.168.10.0 netmask 255.255.255.0 eth0
[root@linux ~]# route add -host 192.168.10.20 eth0
[root@linux ~]# route add -host 192.168.10.30 eth0
[root@linux ~]# route add -host 192.168.10.40 eth0
# 这样就设定妥当啦！将你的路由规划好啰！

[root@linux ~]# route -n
Kernel IP routing table
Destination     Gateway   Genmask         Flags Metric Ref    Use Iface
192.168.10.20   0.0.0.0   255.255.255.255 UH    0      0        0 eth0
192.168.10.30   0.0.0.0   255.255.255.255 UH    0      0        0 eth0
192.168.10.40   0.0.0.0   255.255.255.255 UH    0      0        0 eth0
192.168.10.0    0.0.0.0   255.255.255.0   U     0      0        0 eth1
# 看到上面这一行，虽然我的两块网卡都是在 192.168.10.0/24 ，
# 不过真正针对整个网域传送的，仅有 eth1 那一块的意思！
```

瞧！这样一来，你的 PC1 就可以 ping 到 PC2~PC4 的主机了！数据的传输上面也没有问题。 这个作法是相当有帮助的呐！对于预设架构不想更动的环境来说。 ^_^！ 不过，由这个案例你也可以清楚的知道，能不能联机其实与路由的关系才大哩！ 而路由是双向的，你必须要考虑到这个封包如何回来的问题喔！

重点回顾

- 网络卡的代号为 eth0, eth1, eth2...，而第一张网络卡的第一个虚拟接口为 eth0:0 ...
- 网络卡的参数可使用 ifconfig 直接设定，亦可使用设定档如 /etc/sysconfig/network-scripts/ifcfg-ethn 来设定；
- 路由是双向的，所以由网络封包发送处发送到目标的路由规划，必须要考虑回程时是否具有相对的路由， 否则该封包可能会『遗失』；
- 每部主机都有自己的路由表，此路由表 (routing table) 是作为封包传送时的路径依据；
- 每部可对外 Internet 传送封包的主机，其路由信息中应有一个预设路由 (default gateway)；
- 要让 Linux 作为 Router 最重要的是启动核心的 IP Forward 功能；

- 重复路由可能会让你的网络封包传递到错误的方向；
- 动态路由通常是用在两个 Router 之间沟通彼此的路由规则用的，常见的 Linux 上的动态路由套件为 zebra ；
- arp proxy 可以透过 arp 与 route 的功能，让路由器两端都在同一个网段内；
- 一般来说，路由器上都会有两个以上的网络接口
- 事实上，Router 除了作为路由转换之外，在 Router 上面架设防火墙，亦可在企业内部再分隔出多个需要安全 (Security) 的单位数据的区隔！

课后练习

- 请问您如何将您的 eth0 这个接口修改成为 192.168.100.2 在网域 192.168.100.0/25 之内的网络参数内容？

  因为 192.168.100.0/25 的 netmask 为 255.255.255.128 ，所以可以这样做：
  ifconfig eth0 192.168.100.2 netmask 255.255.255.128 up
  这样即可！如果尚须其它的参数，则需要以档案形式来下达，如 vi /etc/sysconfig/network-scripts/ifcfg-eth0，并修改为：
  DEVICE=eth0
  ONBOOT=yes
  BOOTPROTO=static
  IPADDR=192.168.100.2
  NETMASK=255.255.255.128
  NETWORK=192.168.100.0
  BROADCAST=192.168.100.127

- 请手动设定 eth0:1 这个虚拟接口，使成为网络参数： 192.168.200.2, 网域在 192.168.200.0/24。

  ifconfig eth0:1 192.168.200.2 up

- 如何观察路由表？

  route -n 即可查阅！注意到 0.0.0.0 那个目标(default gateway)。

- 如何启动 Linux 的 IP Forward 功能？

  直接以『echo "1" > /proc/sys/net/ipv4/ip_forward 』即可！

- 假设您是一个学校单位的信息管理员，学校内有 200 部计算机，奉上面大头的旨意，必须要将 200 部计算机分为 4 个 Subnet ，请问您应该如何布线(请画出示意图)？而这 4 个 Subnet 的网络参数如何选择(请自行选择)？而是否需要 Router ？如果需要的话，假设每个 Router 仅能有两个网络实体接口，那么该如何布线？(注：不要使用虚拟接口)
- 假设你想要连接到 168.95.1.1 ，那么你该如何判断你经过『多少个』节点？

可以使用 traceroute 168.95.1.1 来分析每个节点的传送信息，也可以透过 ping 168.95.1.1 所回传的那个 ttl 值判断节点数量。

- 万一您的网络有点停顿，发现可能是网络上某个节点出现问题，您应该如何确认是哪一部 Router 出问题？

  就利用 traceroute 吧！

参考数据

- 动态路由套件 Quagga: http://www.quagga.net
- 动态路由套件 zebra: http://www.zebra.org
- 网中人写的『频宽负载平衡』：http://www.study-area.org/tips/multipath.htm
- Ben 哥写的『实作 Linux 动态路由』：http://linux.vbird.org/somepaper/20060714-linux_cisco_route.pdf
- quagga 官方操作文件：http://www.quagga.net/docs/quagga.pdf
- 酷学园的 ARP Proxy：http://phorum.study-area.org/viewtopic.php?t=5619
- 酷学园 ericshei 的 ARP Proxy 分享：http://phorum.study-area.org/viewtopic.php?t=22943

认识网络安全

最近更新日期：2006/09/06

在介绍了『网络基础』、『限制联机 port number』、『网络升级套件』之后，再来准备要上 Internet 了吗？！如果只是想要上 Internet 去浏览，那么自然没有问题，如果是想要对 Internet 开放网络服务，那么最好还是先认识一下网络安全会比较好一些。什么？套件也更新了， port 也关闭了，还需要认识什么网络安全啊？！呵呵！当然啦！因为难保我们的主机不会被新的套件漏洞以及阻断式攻击 (DoS)所困扰啊！在这个章节里面，我们会稍微介绍一些基础的网络防护观念，尤其是系统管理员应该要做的事情呐！

网络封包联机进入主机的流程

在这一章当中，我们要讨论的是，当来自一个网络上的联机要求想进入我们的主机时， 这个网络封包在进入主机实际取得数据的整个流程是如何？了解了整个流程之后， 你才会发现：原来系统操作的基本概念是如此的重要！ 而你也才会了解要如何保护你的主机安全呐！闲话少说，咱们赶紧来瞧一瞧先。

封包进入主机的流程

在网络基础章节当中我们谈到过目前的网络架构主要是 TCP/IP 为主， 而绝大部分的网络联机是双向的，其中又以 TCP 封包为代表。 另外，根据 Server/Client 的联机方向与 TCP/IP 的概念，我们会知道建立一条可靠的网络联机需要一组 Socket Pair 的辅助， 亦即成对的来源与目标之 IP 与 port 啰，以使联机的两端可以顺利的连接到相对的应用软件上。

上面谈到的这些都是属于网络的基础概念，在这里我们要谈的是，那么要让这个 TCP 封包顺利的进入到 Linux 主机上，然后使用 port 所对应的软件来存取系统的档案系统资源时，还得要经过哪些关卡呢？ 举例来说，如果你的 Linux 主机有开启 WWW 的 port 80 网络服务，而 port 80 是由一个名称为 httpd 的程序所启动的，这个程序的设定文件为 httpd.conf ，那么 Client 的联机要进入到你 Linux 主机的 WWW 时， 会经过什么阶段呢？基本上，会经过如下图的几个阶段：

图一、网络封包进入本机的流程顺序

1. 封包过滤防火墙：IP Filtering 或 Net Filter
   要进入 Linux 本机的封包都会先通过 Linux 核心的预设防火墙，就是称为 IP Filter 或 Net Filter 的咚咚，简单的说，就是 iptables 这个软件所提供的防火墙功能。iptables 这个 Linux 预设的防火墙软件可以针对网络封包的 IP, port, MAC, 以及联机状态如 SYN, ACK 等数据进行分析， 以过滤不受欢迎的网络封包呢！举例来说，如果有个 IP 为 aaa.bbb.ccc.ddd 是个恶意网站来源， 那你就可以透过 iptables 抵挡来自该 IP 的网络封包的联机，以达到基本的主机防火墙功能。 这部份我们会在下一章深入了解。

2. 第二层防火墙：TCP Wrappers
   通过 IP Filter 之后，网络封包会开始接受 Super daemons 及 TCP_Wrappers 的检验，那个是什么呢？ 呵呵！说穿了就是 /etc/hosts.allow 与 /etc/hosts.deny 的设定文件功能啰。 这个功能也是针对 TCP 的 Header 进行再次的分析，同样你可以设定一些机制来抵制某些 IP 或 Port ，好让来源端的封包被丢弃或通过检验；

3. 服务 (daemon) 的功能：
   前面这两个动作基本上是 Linux 预设的功能，而这第三个步骤就是属于软件功能了。 举例来说，你可以在 httpd.conf 这个设定档之内规范某些 IP 来源不能使用 httpd 这个服务来取得主机的数据， 那么即使该 IP 通过前面两层的过滤，他依旧无法取得主机的资源喔！但要注意的是，如果 httpd 这支程序本来就有问题的话，那么 client 端将可直接利用 httpd 软件的漏洞来入侵主机，而不需要取得主机内 root 的密码！因此， 要小心这些启动在因特网上面的软件喔！ 所以前一章网络升级套件是很重要的！

4. 使用主机的档案系统资源：
   想一想，你使用浏览器连接到 WWW 主机最主要的目的是什么？当然就是读取主机的 WWW 数据啦！那 WWW 资料是啥？就是档案啊！^_^！所以，最终网络封包其实是要向主机要求档案系统的数据啦。 我们这里假设你要使用 httpd 这支程序来取得系统的档案数据，但 httpd 预设是由一个系统账号名称为 httpd 来启动的，所以：你的网页数据的权限当然就是要让 httpd 这支程序可以读取才行啊！如果你前面三关的设定都 OK ，最终权限设定错误， 使用者依旧无法浏览你的网页数据的。

在这些步骤之外，我们的 Linux 以及相关的软件都可能还会支持登录文件记录的功能， 为了记录历史历程，以方便管理者在未来的错误查询与入侵侦测，良好的分析登录档的习惯是一定要建立的， 尤其是 /var/log/messages 与 /var/log/secure 这些个档案！ 虽然各大主要 Linux distribution 大多有推出适合他们自己的登录档分析套件，例如 CentOS 的 logwatch ，不过毕竟该套件并不见得适合所有的 distributions ，所以鸟哥尝试自己写了一个 logfile.sh 的 shell script，您可以在底下的网址下载该程序：

- http://linux.vbird.org/download/index.php?action=detail&fileid=60

好了，那么根据这些流程，你觉得我们可以如何保护自己的主机呢？

主机能作的保护： 权限设定、套件更新、SELinux

在基础篇里面的前面几章我们谈到很多关于档案权限方面的注意事项， 关于目录最重要的是那个 w(可写入)的权限，至于对档案来说，那个 r(可读取) 也是非常重要的！而由前一小节的图一我们也知道网络服务其实就是提供主机的档案资源给 client 端来查阅就是了。

根据这样的说法，你可以知道，如果你有某些不想要被读取的数据在主机上面的话， 那么将该数据的权限设定为不能被某些网络服务读取的情况， 就能达到最基础的保护了。所以您说档案权限重不重要啊！很重要的！不是吗？

- 权限的重要性

鸟哥常常在上课的时候会开玩笑，说如果你只要下达一个指令，那你的系统就得要重新安装了！ 那就是：『chmod -R 777 /』，这个指令可是『极度危险』的喔！为何呢？因为系统上面本来就有很多需要被保护的数据， 例如 /etc/shadow 以及 /etc/passwd 等，尤其是 shadow 密码档案。虽然里头是加密过的数据，不过别忘了，现在的 PC 速度实在太快了，而网络上又有太多暴力破解密码的软件，如果你的 /etc/shadow 被取得后，嘿嘿！你的密码其实就算『公开了』。那万一你有开放某些网络服务的话， 例如可联机登入的 ssh 服务或 mail 服务，那任何人都可以使用你的主机来登入， 或者是利用你的主机来收你主机上的其它使用者的信，唉！糗大了！

再者，很多朋友在主机上面常常喜欢建立权限为 drwxrwxrwx 的目录来提供使用者上传数据， 这实在是很危险ㄟ！如果使用者的功力够高的话，他可以在网络软件如 httpd 的使用上， 来建立一些危险的 script 在你的 drwxrwxrwx 的目录中，那如果你不小心进入到该目录， 又不小心执行了该恶意使用者所建立的 script ，恭喜您～中标！

另外，如果你是学校老师，为了公平与同学本身的权益起见，你会希望同学们所上传的数据不会被其它同学所窃取。 那么你该如何进行权限的规范？如果单纯的让学生通通上传到单一目录，并且没有指定特殊的权限时， 不但某些同学的数据可能会被窃取与复制，更惨的是，可能数据会被某些恶意同学所删除！ 那可就麻烦了！所以，权限的设定真的很重要啦！

而除了传统的权限之外，事实上目前 Linux 支持一种称为 ACL 的额外权限控制方式， 也支持更强化安全的 SELinux ，这两个小东西我们会在本章的后面部分继续介绍。

- 严格的密码的重要性：

很多使用者为了方便记忆，老是跟系统管理员说：『喂！我的密码可不可以简单一点啊？ 太麻烦的我都记不住！』如果您是那个可怜的系统管理员，你该如何响应？ 如果你大开方便之门，未来可是后患无穷的！举例来说，如果你的 mail server 上面某个使用者账号为 alex 好了，那么他的 email address 将会是：『 alex@your.host.name 』， 那这个使用者由于使用习惯不良，他将他的 mail address 留在 Internet 上，所以很多人都知道这个 address。

知道就知道，会有什么了不起吗？呵呵！了不起的很！如果有个坏家伙，他想要偷偷的收取 alex 的信， 那他就在他的收信软件上面偷偷填上你的主机，然后偷偷输入账号 alex 并且输入密码为 alex ， 如果你真的帮 alex 这个使用者建立同名的密码，哈哈！系啊(请台语发音，谢谢)！这个 alex 永远都收不到他的

信了！

这算还好吶！如果你有开放远程联机登入的服务，那么坏家伙就可以利用 alex 这个账号与密码来登入你的主机， 如果你没有做好权限规划的话，哇！整部主机的数据被偷光光！那可有的瞧的了！ 所以，您说密码不重要吗？我可不认为！

- 套件更新的重要性：

很多朋友由于网络文章的关系，可能会拿比较旧的 Linux distribution 来作为架站的平台， 举例来说，使用 Red Hat 9 来架站的朋友想必还是不少的。如果你真的利用旧的版本来进行网站的架设， 而且还对 Internet 开放服务的话，那么你的主机将会在不到一天的时间内被『绑架』的！ 为什么呢？因为套件软件都是可能有漏洞的，如果你没有补洞的话.....

有些朋友认为：『我的密码设定的严格一点，应该就好了吧？』真的吗？让我们瞧一瞧图一的流程， 第三个步骤是否使用到 httpd 这个程序的功能了，万一这个程序有问题怎么办？ 举例来说，酷学园的朋友曾经在他举办的研讨会当中露一手如何绑架没有修补漏洞的 Linux 系统， 利用的就是 httpd 这个软件的漏洞，整个入侵的过程没有花费一分钟以上！ 而且他取到的可是 root 的权限吶！不是什么阿猫阿狗的喔！而且他完全没有输入任何密码，使用的入侵程序则是由 Internet 上面取得的。

在上头这个例子鸟哥不是要说该朋友的功力，而是要提醒大家，套件修补的重要性！ 要取得破解程序的管道实在太多了，但如果你都有在最短的时间内取得套件的更新的话， 那么至少该破解程序对你的系统就不会生效！你的主机自然就会比较安全些。 而这个问题在所有的操作系统上面都是存在的！ Windows 系统也是每个月必须要推出他们的套件程序修补， 否则一样会被攻击或入侵啊！不过 Linux 的套件漏洞修补要快多了！

- SELinux

在最新的 Linux 2.6 版核心上所发展的 distributions 目前预设都会启动一个名为 SELinux 的核心模块， 这个 SELinux 必须要在开机加载核心时就得要加载，那这个玩意儿是啥咚咚？ SELinux 是 Security Enhanced Linux (安全加强的 Linux) 的缩写， 他并不是一个防火墙的软件，而是一个『针对档案系统权限作更细部规划的一个模块』。

传统的 Linux 权限是分为三种身份 (owner, group, others) 以及三种权限 (r, w, x)， 但事实上，这三种身份的三种权限组合并无法有效的管理所有系统上的 daemon 存取数据时所需要的行为。因此美国国家安全局便发展出这个可以更细部规划档案权限功能的 SELinux 了。

由于 SELinux 主要是进行档案系统的细部权限设定，所以想要使用 SELinux 的配置时， 需要对 Linux 的档案系统以及基础的操作系统概念要很清楚，否则将会使得很多的网络服务无法正确的启用系统资源， 导致你的主机很多服务无法存取系统数据！因此，对于我们刚接触到 Linux 架站的朋友来说， 建议你先关闭 SELinux ，等到两三年后对于 Linux 有很深的概念后， 再来尝试配置 SELinux 这个有趣的咚咚！

也就是说，如果你没有关闭 SELinux 的话，那么你就得要针对 SELinux 进行档案权限的额外配置， 否则你的网络服务就不可能会正常的启动！那么如何关闭 SELinux 呢？你可以这样做：

```
1. 先关闭 /etc/selinux/config 的内容
```

```
[root@linux ~]# vi /etc/selinux/config
# 将底下的设定值改成这样:
SELINUX=disabled

2. 修改开机时 grub 的设定档
[root@linux ~]# vi /boot/grub/menu.lst
.....省略.....
    kernel /boot/vmlinuz-2.6.9 ro root=/dev/hda1 rhgb selinux=0
.....省略.....

3. 重新开机
[root@linux ~]# sync; reboot
```

因为 SELinux 必须要在开机的时候加载，同样的，要卸载也必须要重新开机才行！ 因此，如果你使用的是您 distributions 的预设安装，那么几乎 SELinux 都是预设启动的！ 你可以依据上述的几个步骤将 SELinux 取消后，重新开机即可。如果对于 SELinux 有兴趣的话， 底下的连结可以参考看看：

- http://fedora.redhat.com/docs/selinux-faq-fc5/
- http://selinux.sourceforge.net/
- http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/selinux-guide/index.html

---

主机的细部权限规划：ACL 的使用

在前一小节当中我们提到 Linux 系统的权限是很重要的，偏偏传统的权限仅有三种身份、三种权限而已，配合 chmod, umask, chown, chgrp 等指令来进行使用者与群组相关权限的设定。如果要进行比较复杂的权限设定时， 例如某个目录要开放给某个特定的使用者来使用时，传统的 owner,group,others 的权限方法可能就无法满足了。 不过还好，我们有 ACL 这个玩意儿可以使用！这玩意挺有趣的，底下我们就来谈一谈：

什么是 ACL？

ACL 是 Access Control List 的缩写，主要的目的是在提供传统的 owner,group,others 的 read,write,execute 权限之外的细部权限设定。ACL 可以针对单一使用者， 单一档案或目录来进行 r,w,x 的权限规范，对于需要特殊权限的使用状况非常有帮助。

由于 ACL 是传统的 Unix-like 操作系统权限的额外支持项目，因此要使用 ACL 必须要有档案系统的支持才行。 目前绝大部分的档案系统都有支持 ACL 的功能，包括 ReiserFS, EXT2/EXT3, JFS, XFS 等等。 在 SuSE 这个版本当中，预设是有启动 ACL 控制的，不过在 CentOS 则预设没有启动 ACL。 所以等一下要使用 ACL 的功能时，你必须要先启动你系统 filesystem 的支持才行喔！

那 ACL 主要可以针对哪些方面来控制权限呢？他主要可以针对几个项目：

- 使用者 (user)：可以针对使用者来设定权限；

- 群组 (group)：针对群组为对象来设定其权限；
- 预设属性 (mask)：还可以针对在该目录下在建立新档案/目录时，规范新数据的预设权限；

好了，再来看看如何让你的档案系统可以支持 ACL 吧！

如何启动 ACL

要让你的档案系统支持 ACL 非常的简单！假如要让你的 /home 支持 ACL 的话，可以直接这样做：

```
[root@linux ~]# mount -o remount,acl /home
[root@linux ~]# mount | grep /home
/dev/hda5 on /home type ext3 (rw,acl)
```

看到那个出现的 ACL 了吧！那就对了～如果没有出现这一行，你的档案系统是无法支持 ACL 的， 那下一节的练习您可就无能为力了～那如果想要一开机就让你的档案系统支持 ACL 呢？ 呵呵！修改 /etc/fstab 就对了！将他改成类似底下的模样：

```
[root@linux ~]# vi /etc/fstab
/dev/hda5   /home    ext3    defaults,acl    1 2
```

加入那一段特殊字体的数据，那么下次开机就能够支持 ACL 了！很简单吧！ ^_^

ACL 的设定技巧： getfacl, setfacl

好了，让你的 filesystem 启动 ACL 支持后，接下来该如何设定与观察 ACL 呢？ 很简单，利用这两个指令就可以了：

- getfacl：取得某个档案/目录的 ACL 设定项目；
- setfacl：设定某个目录/档案的 ACL 规范。

先让我们来瞧一瞧 setfacl 如何使用吧！

```
[root@linux ~]# setfacl [-mxdb] 设定值
参数：
-m  ：设定一个 ACL 规范；
-x  ：取消一个 ACL 规范；
-b  ：全部的 ACL 规范都移除；
-d  ：设定预设的 ACL 规范，仅能针对目录使用。
```

最常用的就是那个 -m 的参数啦！用来定义一笔 ACL 的设定规范说。那么 ACL 该如何设定呢？ 不同的使用者、群组与预设权限设定方法有点不同，不过，基本上有底下这三种简易的设定方法：

```
1. 针对使用者
设定值的规范为： u:[使用者账号列表]:[rwx]
例如针对 dmtsai 这个使用者来规范其权限为 rx ，则：
[root@linux ~]# setfacl -m u:dmtsai:rx somefilename
```

```
2. 针对群组来设定
设定值的规范为： g:[群组名]:[rwx]
例如针对 users 这个群组来规范其权限为 rw ，则：
[root@linux ~]# setfacl -m g:users:rw somefilename

3. 针对预设权限来规范，类似 umask 的功能
设定值的规范为： m:[rwx]
例如假设预设权限为 rwx，则：
[root@linux ~]# setfacl -m m:rwx somefilename
```

了解了上面的设定方式后，现在让我们来实际操作一下吧！假设：

- 你已经将 /home 这个独立的 partition 设定了 ACL 的支持了，
- 并且在 /home 底下设定了一个名称为 project 的目录，
- 该目录要给 eric 这个使用者，且属于 users 这个群组，预设权限应该是 770 ；
- 有个使用者账号名称为 jordan ，他属于 jordan 那个群组，但他想要进入到 project 那个目录来工作， 意思是说，jordan 在该目录下需有 w 的权限才行；
- 有个使用者他是其它班级的老师，名称为 tip ，群组名亦为 tip ，他想要进入该目录查阅所有档案数据， 但是不能够进行删除与新增的工作，亦即他不能拥有 w 的权限。

在传统的 Linux 档案权限中，要达成上述的功能时，你得要让 jordan 与 tip 这两个使用者加入 users 那个群组才行，但是 jordan 是希望可以在该目录内工作的，所以他必须要拥有 w 的权限， 而 tip 却仅能读取，所以他不能拥有 w 的权限！哇！如此一来，就无法完成上述的交代事项了！ 此时我们只好透过 ACL 来单独的针对 tip/jordan 这两个使用者来设定他的权限呐！整个流程可以是这样的：

```
1. 建立该目录并规划好权限：
[root@linux ~]# mkdir /home/project
[root@linux ~]# chown eric:users /home/project
[root@linux ~]# chmod 770 /home/project
[root@linux ~]# ls -ld /home/project
drwxrwx---  2 eric users 4096 Sep  5 15:54 /home/project/
# 瞧！已经将需要的目录规划好了！使用者/群组与权限都 OK 了；

2. 建立 jordan 的使用权限(需要有 w)：
[root@linux ~]# cd /home
[root@linux home]# setfacl -m u:jordan:rwx project

[root@linux home]# getfacl project
# file: project    <==前面三行只是指出这个档名的传统 Linux 权限
# owner: eric
# group: users
user::rwx          <==注意看，这是针对『预设使用者』的权限设定；
user:jordan:rwx    <==这是针对 jordan 的权限设定
group::rwx         <==这是针对『预设群组』的权限设定
mask::rwx          <==这玩意儿则是预设属性啦！
```

```
other::---
# 上面这个输出共 8 行我们会在底下详细说明！

[root@linux home]# ls -ld project
drwxrwx---+ 2 eric users 4096 Sep  5 15:54 project
# 看看！多了一个 + 的标志喔！
```

那个 getfacl 指令可以用来取得某个文件名的 ACL 数据啦！至于输出的共 8 行数据你必须要这样看：

- 第 1-3 行：前面三行会显示出这个档案的 Linux 传统属性，包括使用者、群组与档名，预设会用 # 开头作为说明；
- 接下来的每一行的输出会以底下的格式来处理：

```
针对的目标(使用者、群组等):[各种账号列表]:[rwx]
针对的目标主要有：
          user       使用者
          group      群组
          mask       预设权限
          other      非本群组的其它使用者
各种账号列表中，如果没有任何数据，如 user::rwx ，则代表预设使用者账号；
```

  主要有三个字段，用『 : 』来隔开三个字段；

- 第 4 行『user::rwx』：由于使用者列表字段中没有填写任何账号，所以代表这个权限是针对预设使用者， 亦即是这个目录的拥有人 eric 啦，是 eric 的权限为『 rwx 』的意思说！
- 第 5 行『user:jordan:rwx』：使用者 jordan 在这个目录下具有 rwx 的权限的意思啦！
- 第 6 行『group::rwx』：没有填写群组名称，所以同样是预设群组，亦即是那个 users 啰， 该群组的权限为『rwx』啦；
- 第 7 行『mask::rwx』：预设的 mask 为 rwx 的意思，这个 mask 是有用途的！底下会说明。
- 第 8 行『other::---』：指的就是其它的未规定的使用者与群组的权限了

好了，现在 jordan 这位朋友当他进入 /home/project 后，立刻就会拥有 rwx 的权限了！ 而不需要加入 users 这个群组呢！真是很方便吧！太好了！ 另外，你如何知道某个档名具有额外的 ACL 权限呢？可以参考上面最终的输出结果中， 会发现 /home/project 这个目录的权限项目竟然是出现『 drwxrwx---+ 』呢！ 那个多出来的『 + 』就是表示该档名有额外的 ACL 控件目啦！ 那接下来如何处理 tip 呢？同样使用 ACL 来控制：

```
3. 设定 tip 这个使用者的权限数据：
[root@linux home]# setfacl -m u:tip:rx project
[root@linux home]# getfacl project
```

```
# file: project
# owner: eric
# group: users
user::rwx
user:tip:r-x   <==瞧！多出来的咚咚啦！
user:jordan:rwx
group::rwx
mask::rwx
other::---
```

如此一来， tip 这位使用者则仅能进入该目录去读取而已，而无法进行写入的动作呢！ 是否很方便啊！有了 ACL 的控件目后，您就可以将你系统内的有需要使用到特殊权限设定的目录进行细部设定， 让你的系统变的更合理，更安全啊！

- ACL 内的 mask 项目

虽然这样就能够设定好一个 ACL 控件目，不过你还需要了解到在 ACL 内的 mask 所代表的意义喔！ 在上面的那个小案例当中，我们并没有去设定这个 mask，mask 需要与使用者的权限进行逻辑运算 (AND) 后，才是有效的权限呐(effective permission)！

举例来说，如果你觉得你的目录要让所有的人都暂时仅能读取不能写入时，可以将 ACL 内的 mask 设定为 rx 即可， 那其它人就不需要再额外的设定了！看看底下这个例子：

```
[root@linux ~]# cd /home
[root@linux home]# setfacl -m m:rx project
[root@linux home]# getfacl project
# file: project
# owner: eric
# group: users
user::rwx
user:tip:r-x
user:jordan:rwx         #effective:r-x
group::rwx              #effective:r-x
mask::r-x
other::---
```

上面的输出全部都是 getfacl 的输出结果，鸟哥并没有加工啊！ ^_^！ 原本的 jordan 具有『rwx』的权限，而 mask 仅有『r-x』，两者去比较后 『两者都有的权限才会生效，就称为有效权限 (effective permission) 啰』！ 所以，jordan 则仅会有 rx 的权限而已啊！这样对 mask 的用法瞭了吗？

一些常见的攻击手法与主机的保护方式

我们由图一了解到数据传送到本机时所需要经过的几道防线后， 现在您应该比较清楚为何我们常常在基础篇里面一直谈到设定正确的权限可以保护您的主机了吧？ 那么除了前面的谈到的主机基本保护之外， 通常人家是如何攻击你的 Linux 主机呢？底下我们就来谈一谈吧！ 先了解一下人家是如何攻击你的，我们才有办法想到如何防御，您说是吧？！

- 取得账号信息后猜密码：

由于很多人喜欢用自己的名字来作为账号信息，因此账号的取得是很容易的！ 举例来说，如果你的朋友将你的 email address 不小心泄漏出去，例如： dmtsai@your.host.name 之类的样式， 那么人家就会知道你有一部主机，名称为 your.host.name，且在这部主机上面会有一个使用者账号， 账号名称为 dmtsai ，之后这个坏家伙再利用某些特殊软件例如 nmap 来进行你主机的 port scan 之后，嘿嘿！他就可以开始透过你主机有启动的软件功能来猜你这个账号的密码了！

另外，如果你常常观察你的主机登录文件，那你也会发现如果你的主机有启动 Mail server 的服务时， 你的登录档就会常常出现有些怪家伙尝试以一些奇怪的常见账号在试图猜测你的密码， 举例来说像：admin, administrator, webmaster .... 之类的账号，尝试来窃取你的私人信件。 如果你的主机真的有这类的账号，而且这类的账号还没有良好的密码规划，那就容易『中标』！ 唉！真是麻烦！所以我们常讲，系统账号千万不能给予密码，容易被猜密码啊！

这种猜密码的攻击方式算是最早期的入侵模式之一了，攻击者知道你的账号，或者是可以猜出来你的系统有哪些账号， 欠缺的就只是密码而已， 因此他会『很努力的』去猜你的密码，此时，你的密码规划如果不好的话，很容易就被攻击了！ 主机也很容易被绑架啊！所以，良好的密码设置习惯是很重要的。

不过这种攻击方式比较费时，因为目前很多软件都有密码输入次数的限制， 如果连续输入三次密码还不能成功的登入，那该次联机就会被断线！ 所以，这种攻击方式日益减少，目前偶而还会看到就是了！这也是初级 cracker 会使用的方式之一。 那我们要如何保护呢？基本方式是这样的：

- 减少信息的曝光机会：例如不要将 Email Address 随意散布到 Internet 上头；
- 建立较严格的密码设定规则：包括 /etc/shadow, /etc/login.defs 等档案的设定， 建议您可以参考基础篇内的 账号管理那一章来规范你的使用者密码变更时间等等， 如果主机够稳定且不会持续加入某些账号时，也可以考虑使用 chattr 来限制账号 (/etc/passwd, /etc/shadow) 的更改；
- 完善的权限设定：由于这类的攻击方式会取得你的某个使用者账号的登入权限， 所以如果你的系统权限设定得宜的话，那么攻击者也仅能取得一般使用者的权限而已， 对于主机的伤害比较有限啦！所以说，权限设定是重要的；

- 利用系统的程序漏洞『主动』攻击：

由图一里面的第三个步骤中，我们知道如果你的主机有开放网络服务时， 就必须有启动某个网络软件嘛！我们也知道由于软件可能撰写方式的问题，可能产生一些会被 cracker 乱用的臭虫程序代码，而这些臭虫程序代码由于产生问题的大小，有分为 bug (臭虫，可能会造成系统的不稳定或当机) 与 Security (安全问题，程序代码撰写方式会导致系统的使用权限被恶意者所掌握) 等问题。

当程序的问题被公布后，某些较高阶的 cracker 会尝试撰写一些针对这个漏洞的攻击程序代码， 并且将这个程序代码放置到 cracker 常去的网站上面，藉以推销自己的『功力』..... 鸟哥要提醒的是，这种程序代码『是很容易被取得的』。 当更多『盈盈美黛子(台语，闲闲没事干之意)』取得这些程序代码后，他可能会想要『试一试这个攻击程序的威力』， 所以就拿来『扫射』一番，如果你八字比较轻，或者当天星座学家说你比较倒霉时， 可能就会被不小心的攻击到......

这种攻击模式是目前最常见的，因为攻击者只要拿到攻击程序就可以进行攻击了， 『而且由攻击开始到取

得你系统的 root 权限不需要猜密码， 不需要两分钟，就能够立刻入侵成功』，所以『盈盈美黛子』们最爱的就是这个咚咚了。 但这个玩意儿本身是靠『你主机的程序漏洞』来攻击的，所以，如果你的主机随时保持在实时更新的阶段， 或者是关闭大部分不需要的程序，那就可以躲避过这个问题。因此，你应该要这样做：

- 关闭不需要的网络服务：开的 port 越少，可以被入侵的管道越少， 一部主机负责的服务越单纯，越容易找出问题点。看看前面谈到的 限制 Linux 的联机埠口 一章吧！
- 随时保持更新：这个没话讲！一定要进行的！参考前一章 网络升级套件。
- 关闭不需要的软件功能：举例来说，后面会提到的远程登入服务器 SSH 可以提供 root 由远程登入，那么危险的事情当然要给他取消啊！^_^

- 利用社交工程作欺骗：

社交工程 (Social Engineering) 指的其实很简单，就是透过人与人的互动来达到『入侵』的目的！ @_@! 人与人的互动可以入侵你的主机？鸟哥在呼咙你吗？当然不是。

近日在台湾的社会你不是常看到某些人会以『退税、中奖、花小钱买贵重物品』等名义来欺骗善良老百姓，让老百姓掏出口袋里的金钱给那些可恶的金光党吗？社交工程也是类似的方法。在大公司里面， 或许你可能会接到这样的电话：『我是人事部门的经理，我的账号为何突然间不能登入了？ 你给我看一看，恩？干脆直接帮我另建一个账号，我告诉你我要的密码是....』。如果你一时不查给他账号密码的话， 你的主机可能就这样被绑走了～

社交工程的欺骗方法多的是，包括使用『好心的 email 通知』、『警告信函』、『中奖单』等等， 在在都是要欺骗你的账号密码，有的则利用钓鱼方式来欺骗你在某些恶意网站上面输入你的账号密码， 很讨厌的啦！那要如何防范呢？

- 追踪对谈者：不要一味的相信对方，你必须要有信心的向上呈报， 不要一时心慌就中了计！
- 不要随意透露账号/密码等信息：最好不要随意在 Internet 上面填写这些数据， 真的很危险的！因为在 Internet 上面，你永远不知道对方屏幕前面坐着的是谁？

- 利用程序功能的『被动』攻击：

啥？除了主动攻击之外，还有所谓的被动攻击喔？没错啊，『系金飞』！那如何作被动攻击呢？ 那就得要由『恶意网站』讲起了。如果你喜欢上网随意浏览的话，那么有的时候可能会连上一些广告很多， 或者是一堆弹出式窗口的网站，这些网站有时还会很好心的『提供你很多好用的软件自动下载与安装』的功能， 如果该网站是你所信任的，例如 Red Hat, CentOS, Windows 官网的话，那还好， 如果是一个你也不清楚他是干嘛的网站，那你是否要同意下载安装该软件？

如果你常常在注意一些网络危机处理的相关新闻时，常会发现 Windows 的浏览器 (IE) 有问题， 有时则是全部的浏览器 (Firefox, Netscap, IE...) 都会出现问题。那你会不会觉得奇怪啊， 怎么『浏览器也会有问题？』这是因为很多浏览器会主动的答应对方 WWW 主机所提供的各项程序功能， 或者是自动安装来自对方主机的软件，有时浏览器还可能由于程序发生安全问题， 让对方 WWW 浏览器得以传送恶意程序代码给你的主机来执行，嘿嘿！中标！

那你又会想啊，那我干嘛浏览那样的恶意网站？喝！总是会有些粗心大意的时候啊！ 如果你今天不小心收

到一个 email ，里面告诉你你的银行账号有问题， 希望你赶紧连上某个网页去看看你的账号是否在有问题的行列中，你会不会去？ 如果今天有个网络消息说某某网页在提供大特价商品，那你会不会去碰碰运气？ 都是可能的啊！不过，这也就很容易被对方攻击到了。

那如何防备啊？当然建立良好的习惯最重要了：

- 随时更新主机上的所有套件：如果你的浏览器是没有问题的， 那对方传递恶意程序代码时，你的浏览器就不会执行，那自然安全的多啊！
- 较小化软件的功能：举例来说，让你的收信软件不要主动的下载档案， 让你的浏览器在安装某些软件时，要通过你的确认后才安装，这样就比较容易克服一些小麻烦；
- 不要连接到不明的主机：其实鸟哥认为这个才最难！ 因为很多时候我们都用 google 在搜寻问题的解决之道啊，那你如何知道对方是否是骗人的？ 所以，前面两点防备还是很重要的！不要以为没有连接上恶意网站就不会有问题啊！

- 蠕虫或木马的 rootkit：

rootkit 意思是说可以取得 root 权限的一群工具组 (kit)，就如同前面主动攻击程序漏洞的方法一样，rootkit 主要也是透过主机的程序漏洞。不过， rootkit 也会透过社交工程让使用者下载、安装 rootkit 软件， 结果让 cracker 得以简单的绑架对方主机啊！

rootkit 除了可以透过上述的方法来进行入侵之外，rootkit 还会伪装或者是进行自我复制， 举例来说，很多的 rootkit 本身就是蠕虫或者是木马间谍程序。蠕虫会让你的主机一直发送封包向外攻击， 结果会让你的网络频宽被吃光光，例如 2001-2003 年间的 Nimda, Code Red 等等；至于木马程序 (Trojan Horse) 则会对你的主机进行开启后门 (开一个 port 来让 cracker 主动的入侵)，结果就是....绑架、绑架、绑架！

rootkit 其实挺不好追踪的，因为很多时候他会主动的去修改系统观察的指令， 包括 ls, top, netstat, ps, who, w, last, find 等等，让你看不到某些有问题的程序， 如此一来，你的 Linux 主机就很容易被当成是跳板了！有够危险！那如何防备呢？

- 不要随意安装不明来源的档案或者是不明网站的档案数据；
- 不要让系统有太多危险的指令：例如 SUID/SGID 的程序， 这些程序很可能会造成使用者不当的使用，而使得木马程序有机可趁！
- 可以定时以 rkhunter 之类的软件来追查：有个网站提供 rootkit 程序的检查，你可以前往下载与分析你的主机：
  http://www.rootkit.nl/projects/rootkit_hunter.html

- DoS 攻击法 (Denial of Service)

这类型的攻击中文翻译成『阻断式攻击』，这种攻击法也很要命，而且方法有很多，最常见的就属 SYN Flood 攻击法了！还记得我们在网络基础里面提到的，当主机接收了一个带有 SYN 的 TCP 封包之后，就会启用对方要求的 port 来等待联机，并且发送出回应封包 (带有 SYN/ACK 旗标的 TCP 封包)，并等待 Client 端的再次回应。

好了，在这个步骤当中我们来想一想，如果 cient 端在发送出 SYN 的封包后，却将来自 Server 端的确

认封包丢弃，那么您的 Server 端就会一直空等，而且 Client 端可以透过软件功能，在短短的时间内持续发送出这样的 SYN 封包，那么您的 Server 就会持续不断的发送确认封包，并且开启大量的 port 在空等～呵呵！等到全部主机的 port 都启用完毕，那么.....系统就挂了！

更可怕的是，通常攻击主机的一方不会只有一部！他会透过 Internet 上面的僵尸主机 (已经成为跳板，但网站主却没有发现的主机) 发动全体攻击，让你的主机在短时间内就立刻挂点。 这种 DoS 的攻击手法比较类似『玉石俱焚』的手段， 他不是入侵您的系统，而是要让您的系统挂点呢！ 最常被用来作为阻断式服务的网络服务就是 WWW 了，因为 WWW 通常得对整个 Internet 开放服务。

这种攻击方法也是最难处理的，因为要嘛就得要系统核心有支持自动抵挡 DoS 攻击的机制， 要嘛您就得要自行撰写侦测软件来判断！真是麻烦啊～而除非您的网站非常大， 并且『得罪不少人』，否则应该不会被 DoS 攻击啦！ ^_^

- 其它：

上面提到的都是比较常见的攻击方法，是还有一些高竿的攻击法啦， 不过那些攻击法都需要有比较高的技术水准，例如 IP 欺骗。他可以欺骗你主机告知该封包来源是来自信任网域， 而且透过封包传送的机制，由攻击的一方持续的主动发送出确认封包与工作指令。 如此一来，你的主机可能就会误判该封包确实有响应，而且是来自内部的主机。

不过我们知道因特网是有路由的，而每部主机在每一个时段的 ACK 确认码都不相同， 所以这个方式要达成可以登入，会比较麻烦，所以说，不太容易发生在我们这些小型主机上面啦！ 不过你还是得要注意一下说：

- 设定规则完善的防火墙：利用 Linux 内建的防火墙软件 iptables 建立较为完善的防火墙，可以防范部分的攻击行为；
- 核心功能：这部份比较复杂，您必须要对系统核心有很深入的了解， 才有办法设定好你的核心网络功能。
- 登录文件与系统监控：你可以透过分析登录文件来了解系统的状况， 另外也可以透过类似 MRTG 之类的监控软件 来实时了解到系统是否有异常，这些工作都是很好的努力方向！

- 主机防护小结语：

要让你的系统更安全，没有『三两三』是没办法达成的！我们也一直鼓吹， 『维护网站比架设网站还要重要』的观念！因为『一人得道鸡犬升天』，同样的道理：『一人中标全员挂点』， 不要以为你的主机没有啥重要数据，被入侵或被植入木马也没有关系， 因为我们的服务器通常会对内部来源的主机规范的较为宽松，如果你的主机在公司内部， 但是不小心被入侵的话，那么贵公司的服务器是否就会暴露在危险的环境当中了？

另外，在蠕虫很『发达』的年代，我们也会发现只要局域网络里面有一部主机中标， 整个局域网络就会无法使用网络了，因为频宽已经被蠕虫塞爆！ 如果老板发现他今天没有办法收信了，但无法收信的原因并非服务器挂点， 而是因为内部人员的某部个人计算机中了蠕虫，而那部主机中蠕虫的原因只是因为该使用者不小心去看了一下色情网站， 你觉得老板会高兴的跟该员工一起看色情网站还是 fire 掉该人员？

所以啊，主机防护还是很重要的！不要小看了！提供几个方向给大家思考看看吧：

1. 建立完善的登入密码规则限制;
2. 完善的主机权限设定;
3. 设定自动升级与修补套件漏洞、及移除危险套件;
4. 在每项系统服务的设定当中，强化安全设定的项目;
5. 利用 iptables, TCP_Wrappers 强化网络防火墙;
6. 利用主机监控软件如 MRTG 与 logwatch 来分析主机状况与登录文件;

被入侵后的修复工作

如果你的主机被入侵的话，而你也由于了解到主机监控的需要，所以在最短的时间内发现此一事件， 那么该如何针对这个被入侵的主机来修复？那如果你要修复的话，你这个网管人员还需要哪些额外的技能？ 底下我们就来谈一谈。

网管人员的额外技巧与任务

从前一小节的分析当中，您会发现网管还真的是挺累的，他需要对于操作系统有一定程度的熟悉， 对于程序的运作 (process) 与权限概念，则需要更了解！否则就麻烦了！那除了操作系统的基本概念之外， 咱们网管还需要啥特殊技巧呢？当然需要啊！其实一部主机最常发生问题的状况， 都是由『内部的网络误用所产生的』，所以啊，你只管好主机而已是『没有办法杜绝问题』的啦！ 底下就来谈谈你还需要啥技巧呢？

- 了解什么是需要保护的内容:

我的天呐，还要知道什么是需要保护的呀？呵呵！没错，就是如此！由刚刚我们知道的主机入侵方法当中，不难了解，只要有人坐在您的主机前面，那么任何事都有可能会发生！因此，如果您的主机相当的重要， 请『不要让任何人靠近！』您可以参考一下汤姆克鲁斯在『不可能的任务』里面要窃取一部计算机内的数据的困难度！ ^_^″″

  - 硬件：能锁就锁吧！
  - 软件：还包含最重要的数据呢！

- 预防黑客(Black hats)的入侵:

这可不是开玩笑的，什么是黑客呀！这是因为原本在西部电影当中，坏人都是戴黑色帽子的， 所以之前的人们就称网络攻击者为 Black hats 啦！在预防这方面的攻击者时，除了严格管制网络的登入之外， 还需要特别控制原本您的主机中的人物！就我们小网站来说，不要以为好朋友就随便他啦！ 他说要指定密码是跟他的账号相同比较好记，您就答应他！等到人家用他的密码登入您的主机，并破坏您的主机， 那可就得不偿失了！如果是大企业的话，那么员工使用网络时，也要分等级的呢！ ^_^

- 主机环境安全化:

没什么好讲的，除了多关心，还是多关心！仔细的分析登录档，常常上网看看最新的安全通告，这都是最基础的！ 还包含了以最快的速度更新有问题的套件！因为，越快更新您的套件，就越快可以杜绝黑客的入侵！

- 防火墙规则的订定:

这部份比较麻烦一些啦!因为您必需要不断的测试测试再测试!以取得最佳化的网络安全设定! 怎么说呢？要晓得的是，如果您的防火墙规则订定得太多的时候， 那么一个资料封包就要经过越多的关卡才能完整的

通过防火墙，以进入到主机内部！嘿嘿！ 这可是相当的花费时间的！会造成主机的效能不彰！特别留意这一点呢！

- 实时维护您的主机：

就像刚刚说的，您必需要随时维护您的主机，因为，防火墙不是一经设定之后就不用在再他了！ 因为，再严密的防火墙，也会有漏洞的！这些漏洞包括防火规则设定不良、利用较新的侦测入侵技术、 利用您的旧软件的服务漏洞等等！所以，必需要实时维护您的主机呀！这方面除了分析 log files 之外，也可以藉由实时侦测来进行这个工作！例如 PortSentry 就是蛮不错的一套软件呢！

- 良好的教育训练课程：

不是所有的人都是计算机网络高手，尤其虽然现在信息爆炸但是仍然有很多的机会会遇到计算机白痴呀！这个时候，要晓得的是，我们对于内部网域通常没有太多的规范，那如果他用内部的计算机去做坏事怎么办？ 有时候还是无心的～挖哩～所以说，需要特别的教育训练课程呀！这也是公司需要网管的主因之一！

- 完善的备份计划：

天有不测风云，人有旦夕祸福呀！什么人都不知道什么时候会有大地震、我们也都不知道什么时候会突然的硬盘挂掉去～ 所以说，完善的备份计划是相当重要的！此外，大概没有人会说他的主机是 100% 的安全吧！ 那如果你的系统被入侵，造成数据的损毁时，你要如何复原你的主机啊？呵呵！一个良好的网站管理人员， 无时无刻都会进行重要数据的备份的！很重要啊！ 这一部份请参考一下基础学习篇之 Linux 主机备份的内容吧！ 本书后续的远程联机服务器 SSH 章节内也会提到一个很棒的 rsync 工具，您可以瞧瞧！

入侵恢复工作

所谓『百密一疏』啊，人不是神，总会有考虑不周的情况，万一您的主机就因为这『一疏』导致被入侵了，那该怎么办？由上面的说明当中，我们知道『木马』是很严重的，因为他会在您的系统下开个后门(Back door)让攻击者可以登入您的主机，而且还会窜改您 Linux 上面的程序，让您找不到该木马程序！怎么办？

很多朋友都习惯『反正只要将 root 的密码改回来就好了』 这样的观点，事实上，那样一部主机还是有被做为中继站的危险啊！所以， 万一您的主机被入侵了，最好的方法还是『重新安装 Linux 』会比较干净！

那该如何重新安装呢？很多朋友一再地安装，却一再地被入侵～为什么呢？因为他没有『记取教训』啊！呵呵！ 底下我们就来谈一谈，一部被入侵的主机应该如何修复比较好？

1. 立即拔除网络线：

   既然发现被入侵了，那么第一件事情就是拿掉网络功能！拿掉网络功能最简单的作法自然就是拔掉网络线了！ 事实上，拿掉网络线最主要的功能除了保护自己之外，还可以保护同网域的其它主机。怎么说呢？举个最近 (2003/08) 发病的疾风病毒好了，他会感染同网域之内的其它主机喔！所以，拔除网络线之后， 远程的攻击者立即就无法进入您的 Linux 主机，而且您还可以保护网域内的其它相关主机啊！

2. 分析登录文件信息，搜寻可能的入侵途径：

   被入侵之后，决不是只要重新安装就好，还需要额外分析 『为什么我的主机这一次会被入侵，对方是如何入侵的？』， 如果您能够找出问题点，那么不但您的 Linux 功力立刻增强了，主机也

会越来越安全喔！ 而如果您不知道如何找出被入侵的可能途径，那么重新安装后，下次还是可能被以同样的方法入侵啊！ 粉麻烦的啦！好了，那该如何找出入侵的途径呢？

- o 分析登录档：低级的 cracker 通常仅是利用工具软件来入侵您的系统，所以我们可以藉由分析一些主要的登录档来找出对方的 IP 以及可能有问题的漏洞。可以分析 /var/log/messages, /var/log/secure 还有利用 last 指令来找出上次登入者的信息。
- o 检查主机开放的服务：很多 Linux 使用者常常不晓得自己的系统上面开了多少的服务？ 我们说过， 每个服务都有其漏洞或者是不应该启用的增强型或者是测试型功能，所以，找出您系统上面的服务， 并且检查一下每个服务是否有漏洞，或者是在设定上面有了缺失，然后一个一个的整理吧！
- o 查询 Internet 上面的安全通报： 透过安全通报来了解一下最新的漏洞信息，说不定您的问题就在上面！

3. 重要数据备份：

主机被入侵后，显得问题相当的严重，为什么呢？因为主机上面有相当重要的数据啊！ 如果主机上面没有重要的数据，那么直接重新安装就好了！所以，被入侵之后，检查完了入侵途径， 再来就是要备份重要的数据了。好了，问个问题，什么是『重要数据』？ who, ps, ls 等等指令是重要数据吗？还是 httpd.conf 等设定文件是重要数据？又或者是 /etc/passwd, /etc/shadow 才是重要数据？

呵呵！基本上，重要的数据应该是『非 Linux 系统上面原有的数据』，例如 /etc/passwd, /etc/shadow, WWW 网页的数据, /home 里面的使用者重要档案等等，至于 /etc/*, /usr/, /var 等目录下的数据，就不见得需要备份了。 注意：不要备份一些 binary 执行文件，因为 Linux 系统安装完毕后本来就有这些档案，此外， 这些档案也很有可能『已经被窜改过了』，那备份这些数据，反而造成下次系统还是不干净！

4. 重新全新安装：

备份完了数据，再来就是重新安装 Linux 系统了。而在这次的安装中， 您最好选择适合您自己的安装套件即可，不要全部套件都给他安装上去啊！挺危险的！

5. 套件的漏洞修补：

记得啊，重新安装完毕之后，请立即更新您的系统套件，否则还是会被入侵的啦！鸟哥喜欢先在其它比较干净的环境下将 Internet 上面的漏洞修补套件下载下来，然后烧录起来，然后拿到自己的刚刚安装完成的系统上面，mount CD 之后全部给他更新，更新之后，并且设定了相关的防火墙机制，同时进行下一步骤『 关闭或移除不需要的服务』后，我才将网络线插上主机的网络卡上！因为鸟哥不敢确定在安装完毕后，连上 Internet 去更新套件的这段时间，会不会又受到入侵攻击说....

6. 关闭或移除不需要的服务：

   这个重要性不需要再讲了吧？！启用越少的服务，系统当然可以被入侵的可能性就比较低。

7. 数据回复与恢复服务设定：

   刚刚备份的数据要赶紧的复制回来系统，同时将系统的服务再次的重新开放，请注意， 这些服务的设定最好能够再次的确认一下，避免一些不恰当的设定参数在里头喔！

8. 连上 Internet：

   所有的工作都进行的差不多了，那么才将刚刚拿掉的网络线接上来吧！恢复主机的运作了！

经过这一连串的动作后，您的主机应该会恢复到比较干净的环境，此时还不能掉以轻心， 最好还是参考防火墙的设定，并且多方面的参考 Internet 上面一些老手的经验，好让您的主机可以更安全一些！

重点回顾

- 要管制登入服务器的来源主机，得要了解网络封包的特性，这主要包括 TCP/IP 的封包协议， 以及重要的 Socket Pair ，亦即来源与目标的 IP 与 port 等。在 TCP 封包方面，则还得了解 SYN/ACK 等封包状态；
- TCP 封包要进入我们 Linux 本机，至少需要通过 IP Filter, super daemon/TCP Wrappers, Daemons, 密码验证功能 等等步骤；
- 主机的基本保护之一，就是拥有正确的权限设定。而复杂的权限设定可以利用 ACL 或者是 SELinux 来辅助；
- ACL 必须要让 Filesystem 支持，故可以在 /etc/fstab 内加入 acl 的控制参数；
- 关闭 SELinux 可在 /etc/selinux/config 档案内设定，亦可在核心功能中加入 selinux=0 的项目；
- ACL 主要可针对 user, group, mask 来设定，可针对单一个人账号设定权限；
- 设定 ACL 的方法为使用 setfacl, 查阅则以 getfacl 指令来动作；
- ACL 内的 mask 是很重要的，必须与使用者的权限进行逻辑 AND 的运算，才会得到正确的最终权限；
- rootkit 为一种取得 root 的工具组，您可以利用 rkhunter 来查询您主机是否被植入 rootkit；
- 网管人员应该注意在员工的教育训练还有主机的完善备份方案上面；
- 一些所谓的黑客软件，几乎都是透过您的 Linux 上面的套件漏洞来攻击 Linux 主机的；
- 套件升级是预防被入侵的最有效方法之一；
- 良好的登录档分析习惯可以在短时间内发现系统的漏洞，并加以修复。

课后练习

- 我老是发现我的系统怪怪的，似乎有点停顿的模样，怀疑可能是 CPU 负荷太大，所以要去检查一下系统相关的信息。请问，我该以什么指令去检查我的系统相关的信息？

可以使用 top, sar, free, ps -aux, uptime, last 等功能去查询系统的相关信息喔！然后再以 kill 之类的指令删除；

- 我怀疑我的系统上面有过多的具有 SUID 的档案存在，导致一般使用者可以随意的取得 root 的权限，请问，我要如何找出这些具有 SUID 权限的档案？

  因为 SUID 是 4000 这个权限的模样，所以我可以这样做：
  find / -perm +4000

- 我由国内一些 ftp 网站上下载了 Red Hat 公司释出的套件，我想安装他，但又不知道该套件档案是否被修改过！ 请问我该如何确定这个套件的可用性？

  利用最简易的 MD5 编码来测试一下，例如『 md5sum 套件名称』，再比对与原始套件释出的 MD5 数据是否相同！？

- 如果我发现使用『 setfacl -m u:dmtsai:rwx /path/to/file 』时，系统却显示『setfacl: Operation not supported』， 你认为是哪里出问题？

  这是由于您的 filesystem 没有启用 ACL 支持，或者是系统的核心不支持。 请先使用 mount -o remount,acl /mount_point 测试看能否支持 ACL ，若不支持时，则可能是由于核心版本太旧了。

- 如果要设定 dmtsai 可以使用 /home/project 这个目录 (假设 /home 已经支持 ACL)，在该目录内 dmtsai 可以拥有完整的权限。请问该如何设定该目录？

  除了使用 setfacl -m u:dmtsai:rwx /home/project 之外，还需要设定 setfacl -m m:rwx /home/project ， 因为 ACL 在目录方面，必须透过使用者权限及 mask 的逻辑运算后才能生效！

- SELinux 是否为防火墙？

  SELinux 并非防火墙，他是用来作为更细部权限设定的一个核心模块。

- 良好的密码规划是防备主机的第一要务，请问 Linux 系统当中，关于密码相关的档案与规则设定在哪些档案里面？

  密码的设定规则在 /etc/login.defs 里面！至于密码档案在 /etc/shadow 内！

- 简易说明，当一部主机被入侵之后，应该如何处理？

  找出问题、重新安装、漏洞修补、数据还原！请参考本章最后一节的说明。

Linux 防火墙与 NAT 主机

最近更新日期：2006/09/15

在谈完了基本的网络安全观念之后，这个章节主要就要针对『防火墙』来进行介绍了！目前的防火墙机制主要是以 Linux Kernel 2.6 版的 iptables 为主的，而 iptables 可以使用指令来下达，也可以透过编写 shell script 来进行指令的整合。鸟哥本人比较习惯使用 scripts 来进行 iptables 的机制规划呢！除了 iptables 之外，事实上，比较简单的还有 TCP Wrappers 这个玩意儿，他则主要是针对某些服务来进行管理的呐！本章的内容主要就是在介绍这两个重要的防火墙软件了！

本章的行前准备工作

由于这个章节里面谈到非常多的封包概念，包括 MAC, IP, TCP, UDP, ICMP 等协议， 以及如何抵挡外部 IP 来源的防火墙基础，还有 IP/netmask 的整体网域写法等等。 而鸟哥对于您学习防火墙的建议是希望你可

以使用 shell script 来撰写脚本，如此一来可以让你的防火墙规则比较清晰一点。所以在您开始了解底下的资料之前， 希望你可以先阅读过相关的数据了：

- 已经认识 Shell 以及 Shell script；
- 已经阅读过网络基础那一个章节的内容；
- 已经阅读过前一篇认识网络安全；
- 已经阅读过 路由器那一章节的内容，了解路由的概念；
- 最好拥有两部主机以上的小型局域网络环境，以方便测试防火墙；
- Linux 主机上最好有两张网卡，可以进行多种测试，并架设 NAT 主机；
- 使用 uname -r 确认你的核心是 2.4 或 2.6 版；

若准备妥当了，赶紧来开始进行吧！

认识防火墙

网络安全除了随时注意套件的漏洞，以及网络上的安全通报之外，你最好能够依据自己的环境来订定防火墙机制， 这样对于你的网络环境，会比较有保障一点喔！那么什么是防火墙呢？ 其实防火墙就是在管制进入到我们网域内的主机(或者可以说是网域)的资料封包的一种机制，例如我们在前一章节认识网络安全当中提到的 iptables 就是一种防火墙机制了。当然了，更广义的来说， 只要能够分析与过滤进出我们管理之网域的封包数据，就可以称为防火墙。

而这个防火墙又可以分为硬件防火墙与本机的软件防火墙。硬件防火墙是由厂商设计好的主机硬件， 这部硬件防火墙内的操作系统主要以提供封包数据的过滤机制为主，并将其它的功能拿掉。因为单纯作为防火墙功能而已， 因此封包过滤的速度与效率较佳。至于软件防火墙呢？那就是我们这个章节要来谈论的啊！软件防火墙本身就是在保护系统网络安全的一套软件(或称为机制)，例如 iptables 与 TCP Wrappers 都可以称为软件防火墙。

无论怎么分，反正防火墙就是用来保护我们网络安全的咚咚就对啦！呵呵！我们这个章节主要在介绍 Linux 系统本身提供的软件防火墙的功能，那就是 iptables 。至于 TCP Wrappers 请前往基础篇的 认识系统服务 参考参考喔！

为何需要防火墙

基本上，如果你的系统 (1)已经关闭不需要而且危险的服务； (2)已经将整个系统的所有套件都保持在最新的状态； (3)权限设定妥当且定时进行备份工作； (4)已经教育使用者具有良好的网络、系统操作习惯。那么你的系统实际上已经颇为安全了！要不要架设防火墙？那就见仁见智啰！

不过，毕竟网络的世界是很复杂的，而 Linux 主机也不是一个简单的东西， 说不定哪一天你在进行某个软件的测试时，主机突然间就启动了一个网络服务， 如果你没有管制该服务的使用范围，那么该服务就等于对所有 Internet 开放， 那就麻烦了！因为该服务可能可以允许任何人登入你的系统，那不是挺危险？

所以啰，防火墙能作什么呢？防火墙最大的功能就是帮助你『限制某些服务的存取来源』！ 举例来说： (1)你可以限制档案传输服务 (FTP) 只在子网域内的主机才能够使用，而不对整个 Internet 开放； (2)你可

以限制整部 Linux 主机仅可以接受客户端的 WWW 要求，其它的服务都关闭； (3)你还可以限制整部主机仅能主动对外联机，对我们主机主动联机的封包状态 (TCP 封包的 SYN flag) 就予以抵挡等等。 这些就是最主要的防火墙功能了！

所以鸟哥认为，防火墙最重要的任务就是在规划出：

- 切割被信任(如子网域)与不被信任(如 Internet)的网段；
- 划分出可提供 Internet 的服务与必须受保护的服务；
- 分析出可接受与不可接受的封包状态；

当然啦，咱们 Linux 的 iptables 防火墙软件还可以进行更细部深入的 NAT (Network Address Translation) 的设定，并进行更弹性的 IP 封包伪装功能，不过，对于单一主机的防火墙来说， 最简单的任务还是上面那三项就是了！所以，你需不需要防火墙呢？理论上，当然需要！ 而且你必须要知道『你的系统哪些数据与服务需要保护』，针对需要受保护的服务来设定防火墙的规则吧！ 底下我们先来谈一谈，那在 Linux 上头常见的防火墙类型有哪些？

Linux 系统上防火墙的主要类别

除了以软件及硬件作为防火墙的分类之外，我们也可以使用 防火墙对于数据封包的取得方式来进行分类。 主要可以分为两大类， 分别是代理服务器 (Proxy) 以及 IP Filter。在代理服务器方面， 由名称我们就可以知道，代理服务器仅是代理 Client 端去向 Internet 要求数据，所以 Proxy 其实已经将可代理的协议限制的很少很少，并且由于内部与外部计算机的并不能直接互通， 所以可以达到良好的保护效果；另一种则是上面提到的 IP fileter 啦！利用封包过滤的方式来达到防火墙的目的！

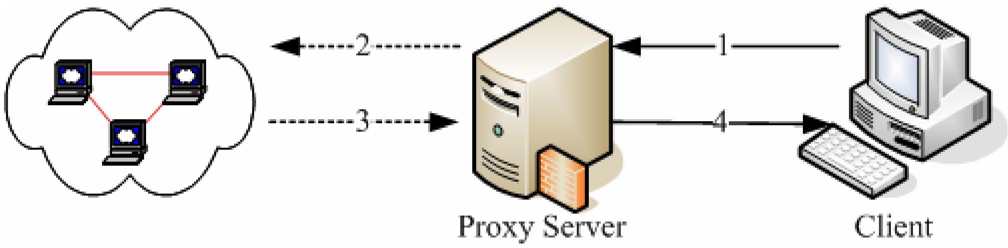
- IP filter (封包过滤机制)

直接使用进入本机的 TCP/IP 上面的封包协议来进行过滤分析，例如利用 TCP/IP 封包表头的 IP 来源、Port number 等数据进行过滤，以判断该封包是否能够进入本机取得本机资源。由于这种方式可以直接分析最底层的封包表头数据， 所以包括硬件地址(MAC), 软件地址 (IP), TCP, UDP, ICMP 等封包的信息都可以进行过滤分析的功能， 因此用途非常的广泛。

在 Linux 上面我们使用核心内建的 iptables 软件来作为防火墙封包过滤的机制， 由于 iptables 是核心内建的功能，因此他的效率非常的高！非常适合于一般小型环境的设定呢！ 他利用一些封包过滤的规则设定，来定义出什么数据可以接收，什么数据需要剔除，以达到保护主机的目的喔！

- Proxy (代理服务器)

其实代理服务器是一种网络服务 (service, daemon)，他可以『代理』使用者的需求， 而代为前往服务器取得相关的资料。就有点像底下这个图示吧：



图一、Proxy Server 的运作原理简介

以上图为例，当 Client 端想要前往 Internet 取得 WWW 的数据时，他取得数据的流程是这样的：

1. 他会向 proxy server 要求数据，请 proxy 帮忙处理；
2. Proxy 可以分析使用者的 IP 来源是否合法？使用者想要去的 WWW 服务器是否合法？ 如果这个 client 的要求都合法的话，那么 Proxy 就会主动的帮忙 client 前往 WWW 服务器取得数据；
3. Internet 所回传的数据是传给 Proxy server 的喔，所以 WWW 服务器上面看到的是 Proxy Server 的 IP 啰；
4. 最后 Proxy 将 client 的要求传回给 client。

这样了解了吗？没错， client 并没有直接连上 Internet ，所以在实线部分(步骤 1, 4)只要 Proxy 与 Client 可以联机就可以了！此时 client 甚至不需要拥有 public IP 哩！而当有人想要攻击 client 端的主机时， 除非他能够攻破 Proxy server ，否则是无法与 client 联机的啦！

另外，一般 proxy 主机通常仅开放 port 80, 21, 20 等 WWW 与 FTP 的埠口而已， 而且通常 Proxy 就架设在 Router 上面，因此可以完整的掌控局域网络内的对外联机！ 让你的 LAN 变的更安全啊！更详细的 Proxy 设定我们会在后续的 代理服务器 章节当中提及的！

在这个章节中，我们先不谈 Proxy 这个东西，而是介绍过滤机制的 iptables 啰！

防火墙的一般线路布线与抵挡技巧

由前面的说明当中，您应该可以了解到一件事，那就是防火墙除了可以『 保护防火墙机制 (iptables) 本身所在的那部主机』之外，还可以『 保护防火墙后面的主机或 PC』。呵呵！ 也就是说，防火墙除了可以防备主机被入侵之外， 他还可以架设在路由器上面藉以控管进出本地端网域 (LAN) 的网络封包。 这种规划对于内部私有网域的安全也有一定程度的保护作用呢！底下我们稍微谈一谈目前常见的防火墙配置吧：

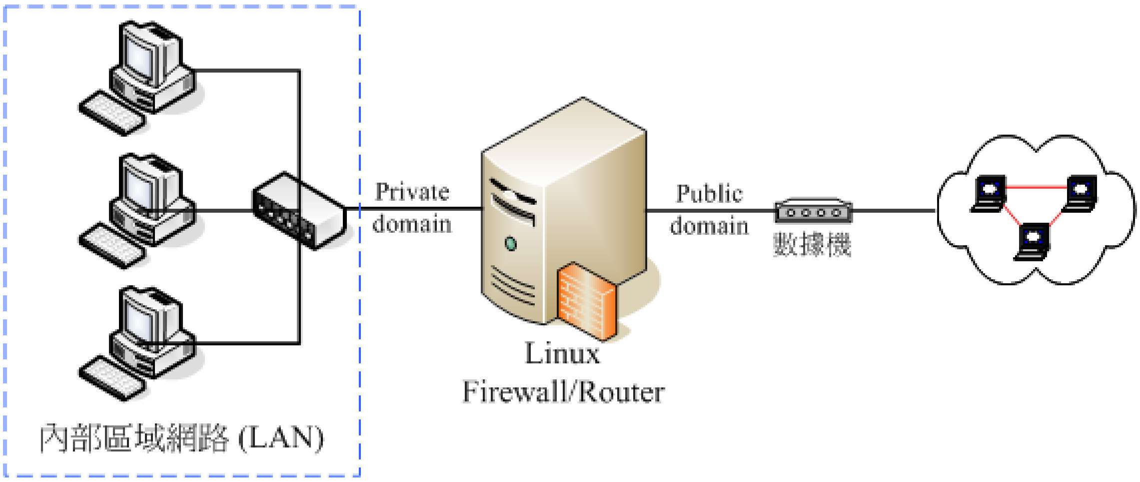
- 单一 Linux 主机兼任防火墙功能：

防火墙除了可以作为 Linux 本机的基本防护之外，他还可以架设在路由器上面以管控整个局域网络的封包进出。 因此，在这类的防火墙上头通常至少需要有两个接口，将可信任的内部与不可信任的 Internet 分开， 所以可以分别设定两块网络接口的防火墙规则啦！整个环境如同下列图二所示。

在图二中，由于防火墙是设定在所有网络封包都会经过的路由器上头， 因此这个防火墙可以很轻易的就掌控到局域网络内的所有封包， 而且你只要管理这部防火墙主机，就可以很轻易的将来自 Internet 的不良网络封包抵挡掉呐。 只要管理一部主机就能够造福整个 LAN 里面的 PC，很划算的啦。

如果你想要将局域网络控管的更严格的话，那你甚至可以在这部 Linux 防火墙上面架设更严格的代理服务器，让客户端仅能连上你所开放的 WWW 服务器而已，而且还可以透过代理服务器的登录文件分析功能，明确的查出来那个使用者在某个时间点曾经连上哪些 WWW 服务器，您瞧瞧！厉害吧！ 如果在这个防火墙上面再加装类似 MRTG 的流量监控软件，还能针对整个网域的流量进行监测。 这样配置的优点是：

- 因为内外网域已经分开，所以安全维护在内部可以开放的权限较大！
- 安全机制的设定可以针对 Linux 主机来维护即可！
- 对外只看的到 Linux 主机，所以对于内部可以达到有效的安全防护！

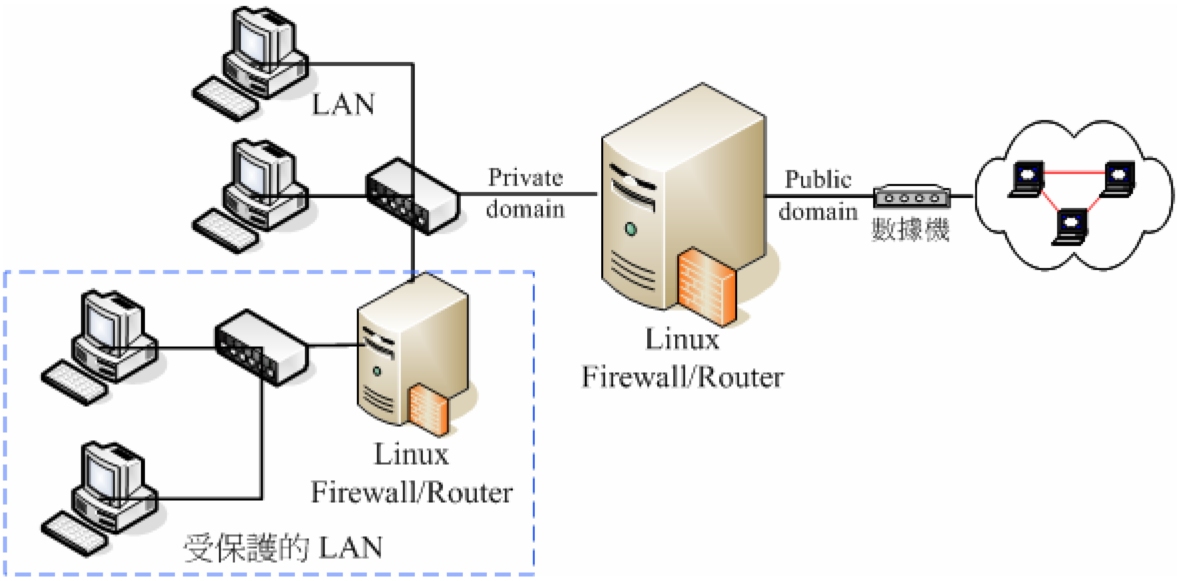


图二、单一 Linux 防火墙主机

- 单一 Linux 防火墙，但 LAN 内另设防火墙

一般来说，我们的防火墙对于 LAN 的防备都不会设定的很严格，因为是我们自己的 LAN 嘛！所以是信任网域之一啰！不过，最常听到的入侵方法也是使用这样的一个信任漏洞！ 因为您不能保证所有使用企业内部计算机的使用者都是公司的员工，也无法保证您的员工不会『搞破坏！』 更多时候是由于某些外来访客利用移动式装置 (笔记型计算机) 连接到公司内部的无线网络来加以窃取企业内部的重要信息。

呵呵！所以，如果您有特别重要的部门需要更安全的保护网络环境，那么将 LAN 里 面再加设一个防火墙，将安全等级分类，那么将会让您的重要数据获得更佳的保护喔！ 整个架构有点像下图三所示。



图三、单一 Linux 防火墙主机，但 LAN 内另设防火墙

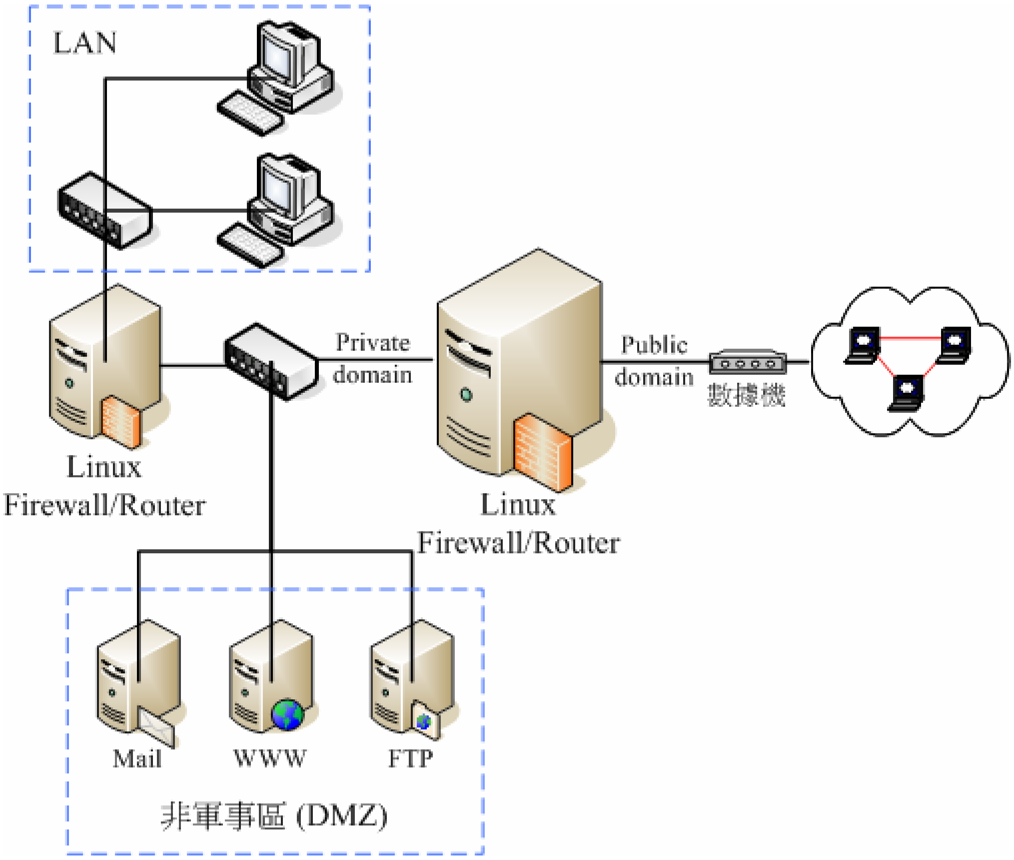
- 在防火墙后端的主机设定

还有一种更有趣的设定，那就是将提供网络服务的服务器放在防火墙后面，这有什么好处呢？ 如下图四所示，Web, Mail 与 FTP 都是透过防火墙连到 Internet 上面去，所以， 底下这四部主机在 Internet 上面的 Public IP 都是一样的！ (这个观念我们会在本章底下的 NAT 主机的时候再次的强调)。 只是透过防火墙的封包分析后，将 WWW 的要求封包转送到 Web 主机，将 Mail 送给 Mail Server 去处理而已(透过 port 的不同来转递)。

好了，因为四部主机在 Internet 上面看到的 IP 都相同，但是事实上却是四部不同的主机， 而当有攻击者想要入侵您的 FTP 主机好了，他使用各种分析方法去进攻的主机，其实是『防火墙』那一部， 攻击者想要攻击您内部的主机，除非他能够成功的搞定您的防火墙，否则就很难入侵您的内部主机呢！

而且，由于主机放置在两部防火墙中间，内部网络如果发生状况时 (例如某些使用者不良操作导致中毒啊、被社交工程攻陷导致内部主机被绑架啊等等的) ，是不会影响到网络主机的正常运作的。 这种方式适用在比较大型的企业当中，因为对这些企业来说，网络主机能否提供正常稳定的服务是很重要的！

不过，这种架构下所进行的设定就得包含 port 的转递，而且要有很强的逻辑概念， 可以厘清封包双向沟通时的流动方式。对于新手来说，设定上有一定的难度， 鸟哥个人不太建议新手这么做，还是等以后有经验之后再来玩这种架构吧！



图四、架设在防火墙后端的主机服务器

通常像上图四的环境中，将网络服务器独立放置在两个防火墙中间的网络，我们称之为非军事区域 (DMZ)。DMZ 的目的就如同前面提到的，重点在保护服务器本身， 所以将 Internet 与 LAN 都隔离开来，如此一来不论是服务器本身，或者是 LAN 被攻陷时， 另一个区块还是完好无缺的！

好了，那么我们 Linux 防火墙软件 iptables 是可以进行封包过滤的，他可以分析网络封包的 socket pair ， 还可以分析不同网络协议的状态，例如 TCP 封包的旗标 (flags) ，甚至可以分析网络卡的卡号呢！ 经由分析这些数据后，咱们的 iptables 至少可以有底下这几种抵挡封包的方式：

- 拒绝让 Internet 的封包进入 Linux 主机的某些 port
  这个应该不难了解吧！例如您的 port 20-21 这个 FTP 相关的 port ， 您只要开放给内部网络的话，所以不对 Internet 开放，那么当 Internet 来的封包想要进入您的 port 20-21 的话，就可以将该数据封包丢掉！因为我们可以分析的到该封包所带有的 port 号码呀！

- 拒绝让某些来源 IP 的封包进入
  例如您已经发现某个 IP 主要都是来自攻击行为的主机，那么只要来自该 IP 的数据封包，就将他丢弃！这样也可以达到基础的安全呦！

- 拒绝让带有某些特殊旗标(flag)的封包进入
  最常拒绝的就是带有 SYN 的主动联机的旗标了！只要一经发现，嘿嘿！您就可以将该封包丢弃呀！

- 分析硬件地址(MAC)来提供服务
  如果您的局域网络里面有比较捣蛋的但是又具有比较高强的网络功力的高手时，如果您使用 IP

来抵挡他使用网络的权限，而他却懂得反正换一个 IP 就好了，都在同一个网域内嘛！ 同样还是在搞破坏～怎么办？没关系，我们可以死锁他的网络卡硬件地址啊！因为 MAC 是焊在网络卡上面的，所以您只要分析到该使用者所使用的 MAC 之后，可以利用防火墙将该 MAC 锁住，呵呵！除非他能够一换再换他的网络卡来取得新的 MAC，否则换 IP 是没有用的啦！

当然还有更多的使用技巧，你可以参考本章最后列出的参考数据， 里头有更多可用的小技巧喔。我们这里仅会真对简单的本机防火墙，以及作为类似 IP 分享器的 NAT 主机作简单的介绍而已啦！ ^_^！好了，开始来玩一玩那个 iptables 吧！

防火墙的使用限制

什么？！设定防火墙之后还不安全啊？！那当然啦！谁说设定了防火墙之后您的系统就一定安全？ 防火墙虽然可以防止不受欢迎的封包进入我们的网络当中，不过，某些情况下， 他并不能保证我们的网络一定就很安全。举几个例子来谈一谈：

- 防火墙并不能很有效的抵挡病毒或木马程序
  假设您已经开放了 WWW 的服务，那么您的 WWW 主机上面，防火墙一定得要将 WWW 服务的 port 开放给 Client 端登入才行吧！否则您的 WWW 主机设定了等于没有用对吧！也就是说，只要进入您的主机的封包是要求 WWW 数据的，就可以通过您的防火墙。那好了，『万一您的 WWW 服务器软件有漏洞，或者本身向您要求 WWW 服务的该封包就是病毒在侦测您的系统』时，您的防火墙可是一点办法也没有啊！ 因为本来设定的规则就是会让他通过啊。

- 防火墙对于来自内部 LAN 的攻击较无承受力
  一般来说，我们对于 LAN 里面的主机都没有什么防火墙的设定，因为是我们自己的 LAN 啊，所以当然就设定为信任网域了！不过， LAN 里面总是可能有些网络小白啊，虽然他们不是故意要搞破坏， 但是他们就是不懂嘛！所以就乱用网络了。这个时候就很糟糕，因为防火墙对于内部的规则设定通常比较少， 所以就容易造成内部员工对于网络误用或滥用的情况。

所以啦，在您的 Linux 主机实地上网之前，还是得先：

- 关闭几个不安全的服务；
- 升级几个可能有问题的套件；
- 架设好最起码的安全防护--防火墙--

其它相关的讯息请到 认识网络安全 里面去看一看怎么增加自身的安全吧！

Linux 的封包过滤机制：iptables

上面谈了这么多，主要还是希望您能了解到防火墙是什么这个议题！而且也希望您知道防火墙并非万能的。好了，那么底下我们终于可以来瞧一瞧，那目前我们的 2.6 版这个 Linux 核心到底使用什么核心功能来进行防火墙设定？

不同 Linux 核心版本的防火墙软件

Linux 的防火墙为什么功能这么好？这是因为他本身就是由 Linux kernel 所提供， 由于直接经过核心来处理，因此效能非常好！不过，不同核心版本所使用的防火墙软件是不一样的！ 因为核心支持的防火墙是逐渐演进来的嘛！

- Version 2.0：使用 ipfwadm 这个防火墙机制；
- Version 2.2：使用的是 ipchains 这个防火墙机制；
- Version 2.4 与 2.6 ：主要是使用 iptables 这个防火墙机制，不过在某些早期的 Version 2.4 版本的 distributions 当中，亦同时支持 ipchains (编译成为模块)，好让使用者仍然可以使用来自 2.2 版的 ipchains 的防火墙规划。不过，不建议在 2.4 以上的核心版本使用 ipchains 喔！

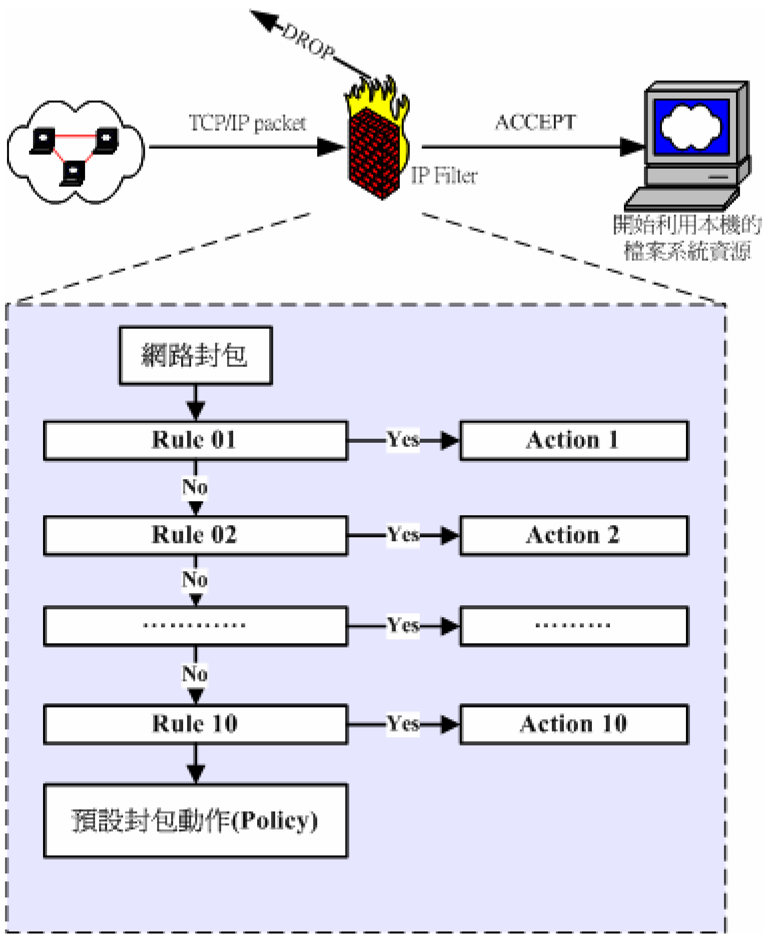
因为不同的核心使用的防火墙机制不同，且支持的软件指令与语法也不相同， 所以在 Linux 上头设定属于你自己的防火墙规则时，要注意啊，先用 uname -r 追踪一下你的核心版本再说！ 如果你是安装 2004 年以后推出的 distributions ，那就不需要担心了，因为这些 distributions 几乎都使用 kernel 2.6 版的核心啊！ ^_^

---

iptables 的表格与封包进入流程

前面的几个小节里面我们一直谈到：『防火墙规则』，咦！啥是规则啊？因为 iptables 是利用封包过滤的机制， 所以他会分析封包的表头数据。根据表头数据与定义的『规则』来决定该封包是否可以进入主机或者是被丢弃。 意思就是说：『根据封包的分析资料 "比对" 你预先定义的规则内容， 若封包数据与规则内容相同则进行动作，否则就继续下一条规则的比对！』 重点在那个『比对与分析顺序』上。

举个简单的例子，假设我预先定义 10 条防火墙规则好了，那么当 Internet 来了一个封包想要进入我的主机， 那么防火墙是如何分析这个封包的呢？我们以底下的图示来说明好了：



图五、封包过滤的规则动作及分析流程

当一个网络封包要进入到主机之前，会先经由 NetFilter 进行检查，那就是 iptables 的规则了。 检查通过则接受 (ACCEPT) 进入本机取得资源，如果检查不通过，则可能予以丢弃 (DROP) ！ 上图五主要的目的在告知您：『规则是有顺序的』！例如当网络封包进入 Rule 1 的比对时， 如果比对结果符合 Rule 1 ，此时这个网络封包就会进行 Action 1 的动作，而不会理会后续的 Rule 2, Rule 3.... 等规则的分析了。

而如果这个封包并不符合 Rule 1 的比对，那就会进入 Rule 2 的比对了！如此一个一个规则去进行比对就是了。 那如果所有的规则都不符合怎办？此时就会透过预设动作 (封包政策, Policy) 来决定这个封包的去向。 所以啦，当您的规则顺序排列错误时，就会产生很严重的错误了。 怎么说呢？让我们看看底下这个例子：

假设您的 Linux 主机提供了 WWW 的服务，那么自然就要针对 port 80 来启用通过的封包规则，但是您发现 IP 来源为 192.168.100.100 老是恶意的尝试入侵您的系统，所以您想要将该 IP 拒绝往来，最后，所有的非 WWW 的封包都给他丢弃，就这三个规则来说，您要如何设定防火墙检验顺序呢？

1. Rule 1 先抵挡 192.168.100.100 ；
2. Rule 2 再让要求 WWW 服务的封包通过；
3. Rule 3 将所有的封包丢弃。

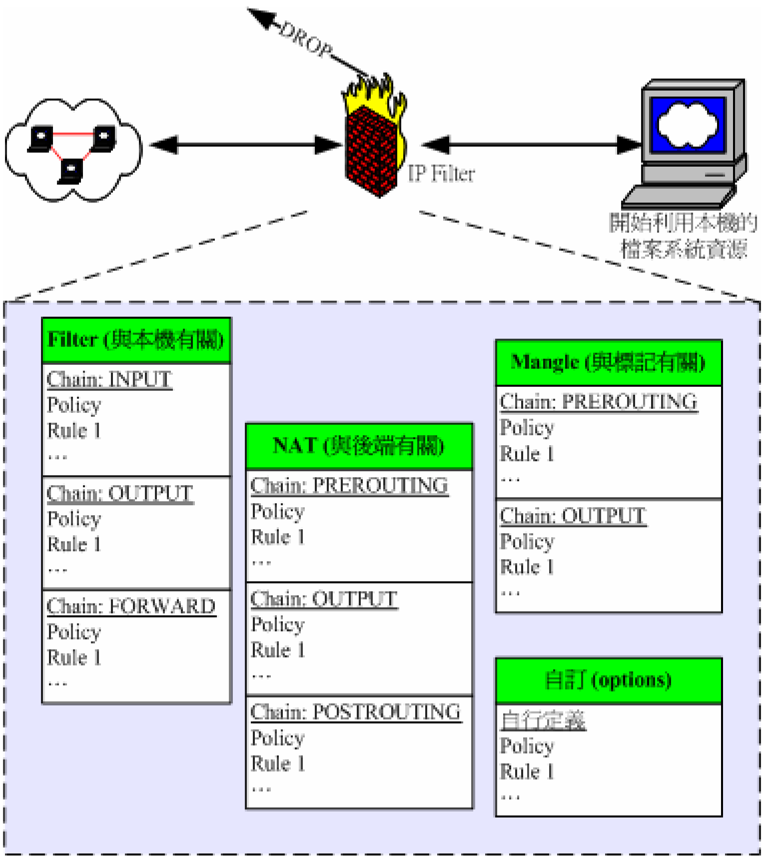
这样的排列顺序就能符合您的需求，不过，万一您的顺序排错了，变成：

1. Rule 1 先让要求 WWW 服务的封包通过；
2. Rule 2 再抵挡 192.168.100.100 ；
3. Rule 3 将所有的封包丢弃。

此时，那个 192.168.100.100 『可以使用您的 WWW 服务』喔！因为只要他对您的主机送出 WWW 要求封包，就可以使用您的 WWW 主机功能了，因为您的规则顺序定义第一条就会让他通过， 而不去考虑第二条规则！这样可以理解规则顺序的意义了吗！现在再来想一想，如果 Rule 1 变成了『将所有的封包丢弃』，Rule 2 才设定『WWW 服务封包通过』，请问，我的 client 可以使用我的 WWW 服务吗？呵呵！答案是『否～』想通了吗？ ^_^

- iptables 的表格与链 (chain)

事实上，那个图五所列出的规则仅是 iptables 众多表格当中的一个链 (chain) 而已。 什么是链呢？这得由 iptables 的名称说起。为什么称为 ip"tables" 呢？ 因为这个防火墙软件里面有多个表格 (table) ，每个表格都定义出自己的预设政策与规则， 且每个表格都用途都不相同。我们可以使用底下这张图来稍微了解一下：



图六、iptables 的表格示意图

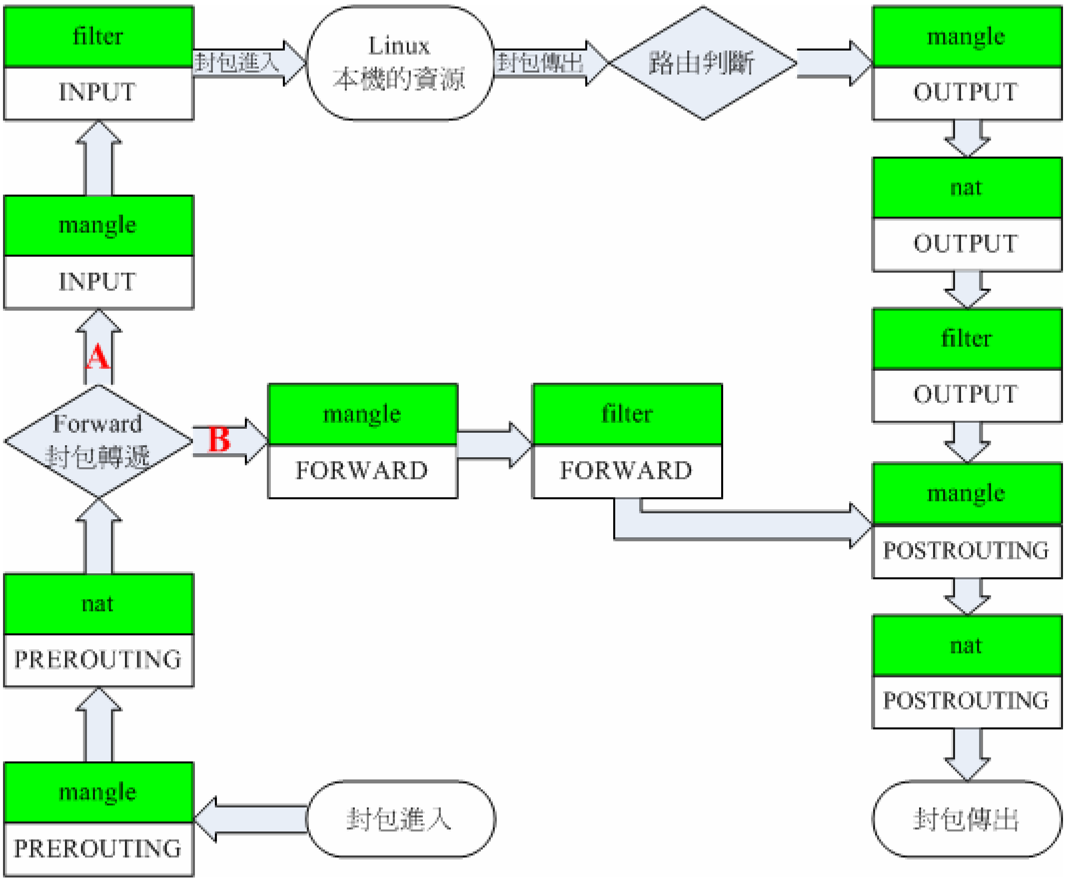
刚刚图五的规则内容仅只是图六内的某个 chain 而已！ 而预设的情况下，咱们 Linux 的 iptables 至少就有三个表格，包括管理本机进出的 filter 、管理后端主机 (防火墙内部的其它计算机) 的 nat 、 管理特殊旗标使用的 mangle (较少使用) 。更有甚者，我们还可以自订额外的链呢！ 真是很神奇吧！每个表格与其中链的用途分别是这样的：

- filter：主要跟 Linux 本机有关，这个是预设的 table 喔！
  - INPUT：主要与封包想要进入我们 Linux 本机有关；
  - OUTPUT：主要与我们 Linux 本机所要送出的封包有关；
  - FORWARD：这个咚咚与 Linux 本机比较没有关系，他可以封包『转递』到后端的计算机中，与 nat 这个 table 相关性很高。

- nat：这个表格主要在用作来源与目的之 IP 或 port 的转换， 与 Linux 本机较无关，主要与 Linux 主机后的局域网络内的计算机较有相关。
  - PREROUTING：在进行路由判断之前所要进行的规则(DNAT/REDIRECT)
  - POSTROUTING：在进行路由判断之后所要进行的规则(SNAT/MASQUERADE)
  - OUTPUT：与发送出去的封包有关

- mangle：这个表格主要是与特殊的封包的路由旗标有关， 早期仅有 PREROUTING 及 OUTPUT 链，不过从 kernel 2.4.18 之后加入了 INPUT 及 FORWARD 链。 由于这个表格与特殊旗标相关性较高，所以像咱们这种单纯的环境当中，较少使用 mangle 这个表格。

那么各个表格与链的相关性可以使用下图来表示：

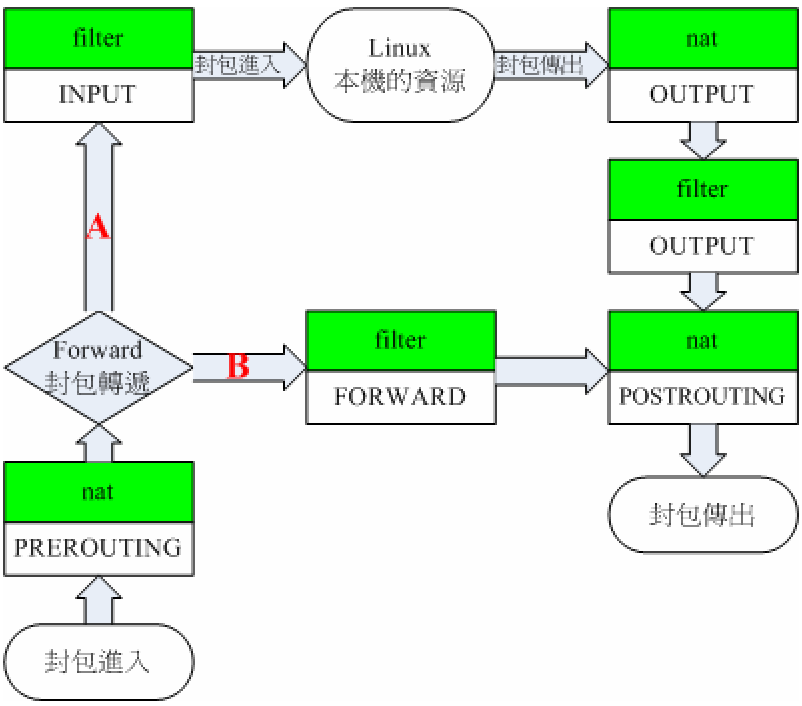


图七、iptables 内建各表格与链的相关性

上面的图示很复杂喔！不过基本上你依旧可以看出来，我们的 iptables 可以控制两种封包的流向：

- 如上图的 A ，封包主要是要读取我们 Linux 本机内的数据，会经过 filter 的 INPUT 链， 而数据的输出则是经过 filter 的 OUTPUT 链；
- 如上图的 B ，封包主要是要透过防火墙而去后端，也就是说，该封包的目标并非我们的 Linux 本机。 主要经过的链是 filter 的 FORWARD 以及 nat 的 POSTROUTING, PREROUTING。

由于 mangle 这个表格很少被使用，如果将图七的 mangle 拿掉的话，那就容易看的多了：



图八、iptables 内建各表格与链的相关性(简图)

透过图八你就可以更轻松的了解到，事实上与本机最有关的其实是 filter 这个表格内的 INPUT 与 OUTPUT 这两条链，如果你的 iptables 只是用来防备 Linux 主机本身的话，那 nat 的规则根本就不需要理他，直接设定为开放即可。

不过，如果你的防火墙事实上是用来管制 LAN 内的其它主机的话，那么你就必须要再针对 filter 的 FORWARD 这条链，还有 nat 的 PREROUTING, POSTROUTING 以及 OUTPUT 进行额外的规则订定才行。 nat 表格的使用需要很清晰的路由概念才能够设定的好，建议新手先不要碰！ 最多就是先玩一玩最阳春的 nat 功能『IP 分享器的功能』就好了！ ^_^！ 这部份我们在本章的最后一小节会介绍的啦！

本机的 iptables 语法

理论上，当你安装好 Linux 之后，系统应该会主动的帮你启动一个阳春的防火墙规则才是。 不过如果您是依照鸟哥的建议来安装 Linux 时，那么安装完毕后，你的系统应该是没有防火墙的啦。 另外，某些早期的版本 (例如 Red Hat 9) 本身同时提供 iptables 及 ipchains 这两个防火墙模块， 不过这两个模块是无法同时存在的！所以你仅能启动其中一个，那当然是启动 iptables 才对啊！ 如果不小心启动了 ipchains 的话 (新版的 Linux 都不会有这个困扰) ，那请使用 rmmod 来移除吧！

不过，在开始进行底下的练习之前，鸟哥这里有个很重要的事情要告知一下。 因为 iptables 的指令会将网络封包进行过滤及抵挡的动作，所以 请不要在远程主机上进行防火墙的练习，因为您很有可能一不小心将自己关在家门外！ 尽量在本机前面登入 tty1-tty6 终端机进行练习，否则常常会发生悲剧啊！ 鸟哥以前刚刚在玩 iptables 时，就常常因为不小心规则设定错误，导致常常要请远程的朋友帮忙重新开机...

刚刚提到咱们的 iptables 至少有三个预设的 table (filter, nat, mangle)，较常用的是本机的 filter 表格， 这也是预设表格啦。另一个则是后端主机的 nat 表格，至于 mangle 较少使用，所以这个章节我们并不会讨论 mangle。 由于不同的 table 他们的链不一样，导致使用的指令语法或多或少都有点差异。 在这个小节当中，我们主要将针对 filter 这个预设表格的三条链来做介绍。底下就来玩一玩吧！

Tips:
防火墙的设定主要使用的就是 iptables 这个指令而已。而防火墙是系统管理员的主要任务之一， 且对于系统的影响相当的大，因此『只能让 root 使用 iptables 』，不论是设定还是观察防火墙规则喔！



- 规则的清除与观察

如果你在安装的时候选择没有防火墙的话，那么 iptables 在一开始的时候应该是没有规则的，不过， 可能因为你在安装的时候就有选择系统自动帮您建立防火墙机制，那系统就会有预设的防火墙规则了！ 无论如何，我们先来看看目前本机的防火墙规则是如何吧！

```
[root@linux ~]# iptables [-t tables] [-L] [-nv]
参数：
-t ：后面接 table ，例如 nat 或 filter ，若省略此项目，则使用预设的 filter
-L ：列出目前的 table 的规则
-n ：不进行 IP 与 HOSTNAME 的反查，显示讯息的速度会快很多！
-v ：列出更多的信息，包括通过该规则的封包总位数、相关的网络接口等

范例：列出 filter table 三条链的规则
[root@linux ~]# iptables -L -n
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

范例：列出更多的信息
[root@linux ~]# iptables -L -nv
Chain INPUT (policy ACCEPT 5748 packets, 746K bytes)
 pkts bytes target     prot opt in     out     source               destination
```

```
....底下省略....
```

仔细看到上面表格的输出，因为没有加上 -t 的参数，所以预设就是 filter 这个表格内的 INPUT，OUTPUT，FORWARD 三条链的规则啰。由于没有规则嘛！所以每个链内部的规则都是空的。 同时注意一下，在每个 chain 后面括号内的 policy 项目，那就是『预设动作(政策)』咯！以上面来看， 虽然我们启动了 iptables ，但是我们没有设定规则，然后政策又是 ACCEPT， 所以是『任何封包都会接受』的意思喔！至于如果加上 -v 的参数时， 则连同该规则所通过的封包总位数也会被列出来啊。底下则是 nat 表格的规则项目：

```
[root@linux ~]# iptables -t nat -L -n
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
```

瞧！与 fiter 表格一模一样吧！只是三条链的内容不同啰！要注意啊！ ^_^！ 以后当你设定每一条防火墙的规则时，记得瞧一瞧设定先！好，那如何清除规则？这样做就对了：

```
[root@linux ~]# iptables [-t tables] [-FXZ]
参数：
-F ：清除所有的已订定的规则；
-X ：杀掉所有使用者 "自订" 的 chain (应该说的是 tables ）啰；
-Z ：将所有的 chain 的计数与流量统计都归零

范例：清除本机防火墙 (filter) 的所有规则
[root@linux ~]# iptables -F
[root@linux ~]# iptables -X
[root@linux ~]# iptables -Z
```

由于这三个指令会将本机防火墙的所有规则都清除，但却不会改变预设政策 (policy) ， 所以如果你不是在本机下达这三行指令时，很可能你会被自己挡在家门外 (若 INPUT 设定为 DROP 时)！ 要小心啊！

一般来说，我们在重新定义防火墙的时候，都会先将规则给他清除掉。还记得我们前面谈到的， 防火墙的『规则顺序』是有特殊意义的，所以啰， 当然先清除掉规则，然后一条一条来设定会比较容易一点啦。底下就来谈谈定义预设政策吧！

- 定义预设政策 (policy)

清除规则之后，再接下来就是要设定规则的政策啦！还记得政策指的是什么吗？『 当您的封包不在您设定的规则之内时，则该封包的通过与否，以 Policy 的设定为准』，在本机方面的预设政策中，假设您对于内部的使用用者有信心的话， 那么 filter 内的 INPUT 链方面可以定义的比较严格一点，而 FORWARD 与 OUTPUT 则可以订定的松一些！通常鸟哥都是将 INPUT 的 policy 定义为 DROP 啦，其它两个则定义为 ACCEPT。 至于 nat table 则暂时不理会他。

```
[root@linux ~]# iptables [-t nat] -P [INPUT,OUTPUT,FORWARD] [ACCEPT,DROP]
参数：
-P ：定义政策( Policy )。注意，这个 P 为大写啊！
ACCEPT ：该封包可接受
DROP   ：该封包直接丢弃，不会让 client 端知道为何被丢弃。

范例：将本机的 INPUT 设定为 DROP ，其它设定为 ACCEPT
[root@linux ~]# iptables -P   INPUT DROP
[root@linux ~]# iptables -P  OUTPUT ACCEPT
[root@linux ~]# iptables -P FORWARD ACCEPT
[root@linux ~]# iptables -L -n
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
# 由于 INPUT 设定为 DROP 而又尚未有任何规则，所以上面的输出结果显示：
# 所有的封包都无法进入你的主机！是不通的防火墙设定！(网络联机是双向的)
```

看到输出的结果了吧？INPUT 被修改设定了喔！其它的 nat table 三条链的设定也是一样的，例如：『 iptables -t nat -P PREROUTING ACCEPT 』就设定了 nat table 的 PREROUTING 链为可接受的意思！预设政策设定完毕后，来谈一谈关于封包的基础比对设定吧。

- 封包的基础比对 IP/netmask I/O 装置

开始来进行封包的比对设定吧！我们先由最基础的 IP 与网域的特征谈起，再谈装置（网络卡）的限制等等。

```
[root@linux ~]# iptables [-AI 链] [-io 网络接口] [-p 协议] \
> [-s 来源 IP/网域] [-d 目标 IP/网域] -j [ACCEPT|DROP]
参数：
-AI 链：针对某的链进行规则的 "插入" 或 "累加"
    -A ：新增加一条规则，该规则增加在原本规则的最后面。例如原本已经有四条规则，
         使用 -A 就可以加上第五条规则！
    -I ：插入一条规则。如果没有指定此规则的顺序，预设是插入变成第一条规则。
         例如原本有四条规则，使用 -I 则该规则变成第一条，而原本四条变成 2~5 号
    链 ：有 INPUT, OUTPUT, FORWARD 等，此链名称又与 -io 有关，请看底下。

-io 网络接口：设定封包进出的接口规范
    -i ：封包所进入的那个网络接口，例如 eth0, lo 等接口。需与 INPUT 链配合；
    -o ：封包所传出的那个网络接口，需与 OUTPUT 链配合；
```

```
-p 协定：设定此规则适用于哪种封包格式
   主要的封包格式有： tcp, udp, icmp 及 all 。

-s 来源 IP/网域：设定此规则之封包的来源项目，可指定单纯的 IP 或包括网域，例如：
   IP  ：192.168.0.100
   网域：192.168.0.0/24, 192.168.0.0/255.255.255.0 均可。
   若规范为『不许』时，则加上 ! 即可，例如：
   -s ! 192.168.100.0/24 表示不许 192.168.100.0/24 之封包来源；

-d 目标 IP/网域：同 -s ，只不过这里指的是目标的 IP 或网域。

-j ：后面接动作，主要的动作有接受 (ACCEPT)、丢弃 (DROP) 及记录 (LOG)
```

iptables 的基本参数就如同上面所示的，仅只谈到 IP 、网域与装置等等的信息， 至于 TCP, UDP 封包特有的埠口 (port number) 与状态 (如 SYN 旗标) 则在下小节才会谈到。 好，先让我们来看看最基础的几个规则，例如开放 lo 这个本机的接口以及某个 IP 来源吧！

```
范例一：所有的来自 lo 这个接口的封包，都予以接受
[root@linux ~]# iptables -A INPUT -i lo -j ACCEPT
# 仔细看上面并没有列出 -s, -d 等等的规则，这表示：不论封包来自何处或去到哪里，
# 只要是来自 lo 这个界面，就予以接受！这个观念挺重要的，就是
#『没有设定的规定，则表示该规定完全接受』的意思！例如这个案例当中，
# 关于 -s, -d...等等的参数没有规定时

范例二：目标来自 192.168.0.1 这个 IP 的封包都予以接受
[root@linux ~]# iptables -A INPUT -i eth0 -s 192.168.0.1 -j ACCEPT
# 不管什么封包格式，只要来自 192.168.0.1 就予以接受。

范例三：目标来自 192.168.1.0/24 可接受，但 192.168.1.10 丢弃
[root@linux ~]# iptables -A INPUT -i eth0 -s 192.168.1.10 -j DROP
[root@linux ~]# iptables -A INPUT -i eth0 -s 192.168.1.0/24 -j ACCEPT
# 上述这两个范例很重要啊！因为有点关系！要先丢弃 192.168.1.10 才能接受该网域。

[root@linux ~]# iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  192.168.0.1          0.0.0.0/0
DROP       all  --  192.168.1.100        0.0.0.0/0
ACCEPT     all  --  192.168.1.0/24       0.0.0.0/0
# 瞧！刚刚的设定在这里已经生效啰！
```

这就是最单纯、简单的防火墙规则的设定与观察方式。你在设定完毕后，都可以利用 iptables -L -n 或 iptables -L -v 来简单的查阅一下。而如果你想要记录某个规则的纪录怎么办？可以这样做：

```
[root@linux ~]# iptables -A INPUT -s 192.168.2.200 -j LOG
```

```
[root@linux ~]# iptables -L -n
target prot opt source          destination
LOG     all  --  192.168.2.200  0.0.0.0/0    LOG flags 0 level 4
```

看到输出结果的最左边，会出现的是 LOG 喔！只要有封包来自 192.168.2.200 这个 IP 时， 那么该封包的相关信息就会被写入到核心讯息，亦即是 /var/log/messages 这个档案当中。 然后该封包会继续进行后续的规则比对。所以说， LOG 这个动作仅在进行记录而已，并不会影响到这个封包的其它规则比对的。好了，接下来我们分别来看看 TCP,UDP 以及 ICMP 封包的其它规则比对吧！

- TCP, UDP 的规则比对

我们在网络基础谈过各种不同的封包格式， 在谈到 TCP 与 UDP 时，比较特殊的就是那个埠口 (port number)，在 TCP 方面则另外有所谓的联机封包状态， 包括最常见的 SYN 主动联机的封包格式。那么如何针对这两种封包格式进行防火墙规则的设定呢？你可以这样看：

```
[root@linux ~]# iptables [-AI 链] [-io 网络接口] [-p tcp,udp] \
> [-s 来源 IP/网域] [--sport 埠口范围] \
> [-d 目标 IP/网域] [--dport 埠口范围] -j [ACCEPT|DROP]
参数：
--sport 埠口范围：限制来源的端口口号码，端口口号码可以是连续的，例如 1024:65535
--dport 埠口范围：限制目标的端口口号码。
```

事实上就是多了那个 --sport 及 --dport 这两个玩意儿，重点在那个 port number 上面啦！ 底下让我们来进行几个小测试：

```
范例一：想要联机进入本机 port 21 的封包都抵挡掉：
[root@linux ~]# iptables -A INPUT -i eth0 -p tcp --dport 21 -j DROP

范例二：想连到我这部主机的网芳 (upd port 137,138 tcp port 139,445) 就放行
[root@linux ~]# iptables -A INPUT -i eth0 -p udp --dport 137:138 -j ACCEPT
[root@linux ~]# iptables -A INPUT -i eth0 -p tcp --dport 139 -j ACCEPT
[root@linux ~]# iptables -A INPUT -i eth0 -p tcp --dport 445 -j ACCEPT
```

瞧！你可以利用 UDP 与 TCP 协议所拥有的端口口号码来进行某些服务的开放或关闭喔！ 你还可以综合处理呢！例如：只要来自 192.168.1.0/24 的 1024:65535 埠口的封包， 只要想要联机到本机的 ssh port 就予以抵挡，可以这样做：

```
[root@linux ~]# iptables -A INPUT -i eth0 -p tcp -s 192.168.1.0/24 \
> --sport 1024:65534 --dport ssh -j DROP
```

注意啊！ 如果你有使用到 --sport 及 --dport 的参数时，就必须指定 udp 或 tcp 的封包格式才行！ 否则的话， iptables 的指令就会出现如下的错误：

```
[root@linux ~]# iptables -A INPUT -i eth0 --dport 21 -j DROP
iptables v1.2.11: Unknown arg `--dport'
Try `iptables -h' or 'iptables --help' for more information.
```

你应该会觉得很奇怪，怎么『 --dport 』会是未知的参数 (arg) 呢？这是因为你没有加上 -p tcp 或 -p udp 的缘故啊！因为 port 是 TCP,UDP 特有的，其它类似 ICMP 则没有这种类的端口口数据啊！ 这样说，

您可以理解吧！ ^_^

除了埠口之外，在 TCP 还有特殊的旗标啊！最常见的就是那个主动联机的 SYN 旗标了。 我们在 iptables 里面还支持『 --syn 』的处理方式，我们以底下的例子来说明好了：

```
范例：将来自任何地方来源 port 1:1023 的主动联机到本机端的 1:1023 联机丢弃
[root@linux ~]# iptables -A INPUT -i eth0 -p tcp --sport 1:1023 \
> --dport 1:1023 --syn -j DROP
```

一般来说，client 端启用的 port 都是大于 1024 以上的埠口，而 server 端则是启用小于 1023 以下的埠口在监听的。所以我们可以让来自远程的小于 1023 以下的端口口数据的主动联机都给他丢弃！ 但不适用在 FTP 的主动联机中！这部份我们未来在 FTP 章节当中再来谈吧！

- 状态模块：MAC 与 RELATED

在早期的 kernel 2.2 以前使用 ipchains 管理防火墙时，通常会让系统管理员相当头痛！ 因为 ipchains 没有所谓的封包状态模块，因此我们必须要针对封包的进、出方向进行管控。 举例来说，如果你想要联机到远程主机的 port 22 时，你必须要针对两条规则来设定：

- 本机端的 1024:65535 到远程的 port 22 必须要放行 (OUTPUT 链)；
- 远程主机 port 22 到本机的 1024:65535 必须放行 (INPUT 链)；

这会很麻烦！因为如果你要联机到 10 部主机的 port 22 时，假设 OUTPUT 为预设开启 (ACCEPT)， 你依旧需要填写十行规则，让那十部远程主机的 port 22 可以联机到你的本地端主机上。 那如果开启全部的 port 22 呢？又担心某些恶意主机会主动以 port 22 联机到你的机器上！ 同样的道理，如果你要让本地端主机可以连到外部的 port 80 (WWW 服务)，那就更不得了～ 这就是网络联机是双向的一个很重要的概念！

好在我们的 iptables 免除了这个困扰！他可以透过一个状态模块来分析 『这个想要进入的封包是否为刚刚我发出去的响应？』 如果是刚刚我发出去的响应，那么就可以予以接受放行！哇！真棒！这样就不用管远程主机是否联机进来的问题了！ 那如何达到呢？看看底下的语法：

```
[root@linux ~]# iptables -A INPUT -m state --state 状态
参数：
-m : 一些 iptables 的模块，主要常见的有：
     state : 状态模块
     mac   : 网络卡硬件地址 (hardware address)
--state : 一些封包的状态，主要有：
     INVALID     : 无效的封包，例如数据破损的封包状态
     ESTABLISHED: 已经联机成功的联机状态；
     NEW         : 想要新建立联机的封包状态；
     RELATED     : 这个最常用！表示这个封包是与我们主机发送出去的封包有关

范例：只要已建立或相关封包就予以通过，只要是不合法封包就丢弃
[root@linux ~]# iptables -A INPUT -m state \
> --state RELATED,ESTABLISHED -j ACCEPT
```

```
[root@linux ~]# iptables -A INPUT -m state --state INVALID -j DROP
```

所以说，如果你的 Linux 主机只想要作为 client 的用途，不许所有主动对你联机的来源， 那么你可以这样做即可：

1. 清除所有已经存在的规则 (iptables -F...)
2. 设定预设政策，除了 INPUT 预设为 DROP 其它为预设 ACCEPT；
3. 开放本机的 lo 可以自由放行；
4. 设定有相关的封包状态可以联机进入本机。

这就是最最阳春的防火墙，你可以透过第二步骤抵挡所有远程的来源封包， 而透过第四步骤让你要求的远程主机响应封包可以进入， 加上让本机的 lo 这个内部循环装置可以放行，嘿嘿！一部 client 专用的防火墙规则就 OK 了！ 你可以在某个 script 上面这样做即可：

```
#!/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin; export PATH
iptables -F
iptables -X
iptables -Z
iptables -P   INPUT DROP
iptables -P  OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
#iptables -A INPUT -i eth0 -s 192.168.1.0/24 -j ACCEPT
```

那如果局域网络内有其它的主机时，再将上表最后一行的 # 取消，就可以接受来自本地 LAN 的其它主机的联机了。 而如果你担心某些 LAN 内的恶意来源主机会主动的对你联机时，那你还可以针对信任的本地端主机的 MAC 进行过滤！ 同样是使用状态模块！这次的状态则是 MAC 的比对。举例来说：

```
范例一：针对局域网络内的 aa:bb:cc:dd:ee:ff 主机开放其联机
[root@linux ~]# iptables -A INPUT -m mac --mac-source aa:bb:cc:dd:ee:ff \
>  -j ACCEPT
参数：
--mac-source ：就是来源主机的 MAC 啦！
```

透过这个玩意儿，你就可以定义更严格的 LAN 内的其它主机能否联机到你的主机的权限了！

- ICMP 封包规则的比对

在网络基础的 ICMP 协议当中我们知道 ICMP 的格式相当的多，而且很多 ICMP 封包的类型格式都是为了要用来进行网络检测用的！ 所以最好不要将所有的 ICMP 封包都丢弃！通常我们会把 ICMP type 8 (echo request) 拿掉而已， 让远程主机不知道我们是否存在，也不会接受 ping 的响应就是了。ICMP 封包格式的处理是这样的：

```
[root@linux ~]# iptables -A INPUT -p icmp --icmp-type 类型 -j ACCEPT
参数：
```

```
--icmp-type ：后面必须要接 ICMP 的封包类型，也可以使用代号，
              例如 8  代表 echo request 的意思。

范例：让 0,3,4,11,12,14,16,18 的 ICMP type 可以进入本机：
[root@linux ~]# vi somefile
#!/bin/bash
icmp_type="0 3 4 11 12 14 16 18"
for typeicmp in $icmp_type
do
   iptables -A INPUT -i eth0 -p icmp --icmp-type $typeicmp -j ACCEPT
done

[root@linux ~]# sh  somefile
```

这样就能够开放部分的 ICMP 封包格式进入本机进行网络检测的工作了！真好！不是嘛！^_^

防火墙的记录、回复与测试

刚刚上面我们谈了很多的设定了，那么我该如何观察目前主机上面的防火墙规则呢？ 我们可以使用『iptables -L -n 』来观察，不过，该指令所显示的信息其实还是不太足够的。 这个时候，我们其实可以使用底下的两个指令来将目前主机上面的防火墙机制『储存』下来， 在下次想要将这个规则『回复』的时候，就能够直接利用指令将规则直接回复喔！

```
[root@linux ~]# iptables-save > filename
[root@linux ~]# iptables-restore < filename
```

一个是储存一个是回复！而在 Red Hat 系统的 RHEL,CentOS,Fedora 当中，如果你将那个 filename 档案存成『 /etc/sysconfig/iptables 』，并且利用 chkconfig 将 iptables 在开机时预设启动的话，那么一开机系统就会主动的帮你把防火墙的规则给加载了就是！ 那么使用 iptables-save 所得到的结果会是如何呢？让我们来看看：

```
[root@linux ~]# iptables-save
# Generated by iptables-save v1.2.11 on Mon Sep 11 17:47:35 2006
*filter    <==使用的 table
:INPUT DROP [7335:859454] <==三条预设的链与预设政策
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [16992:13134791]
-A INPUT -i lo -j ACCEPT  <==开始各个规则的设定
-A INPUT -m state --state RELATED -j ACCEPT
-A INPUT -m mac --mac-source 00:04:75:D0:A2:58 -j ACCEPT
-A INPUT -m state --state ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A INPUT -i eth0 -p icmp -m icmp --icmp-type 3 -j ACCEPT
....中间省略....
-A INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
COMMIT
```

```
# Completed on Mon Sep 11 17:47:35 2006
# 井号 (#) 是批注，星号 (*) 代表预设的 table，而冒号 (:) 代表各条链的预设政策;
# 后续的动作则是各个规则啦!
```

你瞧到输出的结果啦！整个数据几乎就是类似手动在指令列模式输入的指令！ 比起 iptables -L -n 所得到的信息要仔细的多。这也是 iptables 的特殊格式， 可以用在 iptables-restore 的指令读入呢！ 比起这种方式，鸟哥还是比较喜欢使用 script 来撰写自己的防火墙规则啦。 制订好规则后当然就是要测试啰！那么如何测试呢？

1. 先由主机向外面主动联机试看看；
2. 再由私有网域内的 PC 向外面主动联机试看看；
3. 最后，由 Internet 上面的主机，主动联机到您的 Linux 主机试看看；

一步一步作下来，看看问题出在哪里，然后多多的去改进、改良！基本上， 网络上目前很多的资料可以提供您不错的参考了！这一篇的设定写的是很简单， 大部分都还在介绍阶段而已！希望对大家有帮助！ 鸟哥在参考数据当中列出几个有用的防火墙网页， 希望大家有空真的要多多的去看看！会很有帮助的！

IPv4 的核心管理功能： /proc/sys/net/ipv4/*

除了 iptables 这个防火墙软件之外，其实咱们 Linux kernel 2.6 提供很多核心预设的攻击抵挡机制喔！由于是核心的网络功能，所以相关的设定数据都是放置在 /proc/sys/net/ipv4/ 这个目录当中。 至于该目录下各个档案的详细资料，可以参考核心的说明文件：

- /usr/src/linux-{version}/networking/ip-sysctl.txt

上面的这个说明数据可以由 http://www.kernel.org 这个网站下载任何一个核心原始码后，解压缩就能够看到。 鸟哥这里也放一份备份：

- http:/linux.vbird.org/linux_server/0250simple_firewall/ip-sysctl.txt

有兴趣的话应该要自行去查一查比较好的喔！我们底下就拿几个简单的档案来作说明吧！

- /proc/sys/net/ipv4/tcp_syncookies

我们在前一章谈到所谓的阻断式服务 (DoS) 攻击法当中的一种方式，就是利用 TCP 封包的 SYN 三向交握原理所达成的， 这种方式称为 SYN Flooding 。那如何预防这种方式的攻击呢？我们可以启用核心的 SYN Cookie 模块啊！ 这个 SYN Cookie 模块可以在系统用来启动随机联机的埠口 (1024:65535) 即将用完时自动启动。

当启动 SYN Cookie 时，主机在发送 SYN/ACK 确认封包前，会要求 Client 端在短时间内回复一个序号，这个序号包含许多原本 SYN 封包内的信息，包括 IP、port 等。若 Client 端可以回复正确的序号， 那么主机就确定该封包为可信的，因此会发送 SYN/ACK 封包，否则就不理会此一封包。

透过此一机制可以大大的降低无效的 SYN 等待埠口，而避免 SYN Flooding 的 DoS 攻击说！ 那么如何启动这个模块呢？很简单，这样做即可：

```
[root@linux ~]# echo "1" > /proc/sys/net/ipv4/tcp_syncookies
```

但是这个设定值由于违反 TCP 的三向交握 (因为主机在发送 SYN/ACK 之前需要先等待 client 的序号响应)， 所以可能会造成某些服务的延迟现象，例如 SMTP (mail server)。 不过总的来说，这个设定值还是不错用的！ 只是不适合用在负载已经很高的服务器内喔！ 因为负载太高的主机有时会让核心误判遭受 SYN Flooding 的攻击呢。

如果是为了系统的 TCP 封包联机最佳化，则可以参考 tcp_max_syn_backlog, tcp_synack_retries, tcp_abort_on_overflow 这几个设定值的意义。

- /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

阻断式服务常见的是 SYN Flooding ，不过，我们知道系统其实可以接受使用 ping 的响应， 而 ping 的封包是可以给很大的！想象一个状况， 如果有个搞破坏的人使用 1000 台主机传送 ping 给你的主机，而且每个 ping 都高达数百 K bytes时， 你的网络频宽会怎样？要嘛就是频宽被吃光，要嘛可能系统会当机！这种方式分别被称为 ping flooding (不断发 ping) 及 ping of death (发送大的 ping 封包)。

那如何避免呢？取消 ICMP 类型 8 的 ICMP 封包回应就是了。我们可以透过防火墙来抵挡， 这也是比较建议的方式。当然也可以让核心自动取消 ping 的响应。不过您必须要了解， 某些局域网络内常见的服务 (例如动态 IP 分配 DHCP 协议) 会使用 ping 的方式来侦测是否有重复的 IP ，所以你最好不要取消所有的 ping 响应比较好。

核心取消 ping 回应的设定值有两个，分别是：/proc/sys/net/ipv4 内的 icmp_echo_ignore_broadcasts (仅有 ping broadcast 地址时才取消 ping 的回应) 及 icmp_echo_ignore_all (全部的 ping 都不回应)。鸟哥建议设定 icmp_echo_ignore_broadcasts 就好了。 你可以这么做：

```
[root@linux ~]# echo "1" >  \
> /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

- /proc/sys/net/ipv4/conf/网络接口/*

咱们的核心还可以针对不同的网络接口进行不一样的参数设定喔！网络接口的相关设定放置在 /proc/sys/net/ipv4/conf/ 当中，每个接口都以接口代号做为其代表，例如 eth0 接口的相关设定数据在 /proc/sys/net/ipv4/conf/eth0/ 内。那么网络接口的设定数据有哪些比较需要注意的呢？ 大概有底下这几个：

- rp_filter：称为逆向路径过滤 (Reverse Path Filtering)， 可以藉由分析网络接口的路由信息配合封包的来源地址，来分析该封包是否为合理。举例来说，你有两张网卡，eth0 为 192.168.10.100/24 ，eth1 为 public IP 。那么当有一个封包自称来自 eth1 ，但是其 IP 来源为 192.168.10.200 ， 那这个封包就不合理，应予以丢弃。这个设定值建议可以启动的。

- log_martians：这个设定数据可以用来启动记录不合法的 IP 来源， 举例来说，包括来源为 0.0.0.0、127.x.x.x、及 Class E 的 IP 来源，因为这些来源的 IP 不应该应用于 Internet 啊。记录的数据预设放置到核心放置的登录档 /var/log/messages。

- accept_source_route：或许某些路由器会启动这个设定值， 不过目前的设备很少使用到这种来源路由，你可以取消这个设定值。

- accept_redirects：当你在同一个实体网域内架设一部路由器，但这个实体网域有两个 IP 网域，例如 192.168.0.0/24，192.168.1.0/24。此时你的 192.168.0.100 想要向 192.168.1.100 传送讯息时，路由器可能会传送一个 ICMP redirect 封包告知 192.168.0.100 直接传送数据给 192.168.1.100 即可，而不需透过路由器。因为 192.168.0.100 与 192.168.1.100 确实是在同一个实体线路上 (两者可以直接互通)，所以路由器会告知来源 IP 使用最短路径去传递数据。但那两部主机在不同的 IP 段，却是无法实际传递讯息的！这个设定也可能会产生一些轻微的安全风险，所以建议关闭他。

- send_redirects：与上一个类似，只是此值为发送一个 ICMP redirect 封包。 同样建议关闭。(事实上，鸟哥在某补教中心教同学架设路由器时，就曾经为了这个 ICMP redirect 的问题伤脑筋！其实关闭 redirect 的这两个项目即可啊！)

要达成上面的功能你必须要这样做：

```
[root@linux ~]# vi somefile
for i in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo "1" > $i
done
for i in /proc/sys/net/ipv4/conf/*/log_martians; do
    echo "1" > $i
done
for i in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo "0" > $i
done
for i in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo "0" > $i
done
for i in /proc/sys/net/ipv4/conf/*/send_redirects; do
    echo "0" > $i
done

[root@linux ~]# sh somefile
```

本机防火墙的一个实例

介绍了这么多的防火墙语法与相关的注意事项后，终于要来架设防火墙了。如同前面谈到的， 你当然可以使用 iptables-save 的语法将相关的防火墙规则转存到 /etc/sysconfig/iptables 去， 然后透过 iptables-restore 或者是重新启动 iptables 来启用你的新防火墙规则。 不过鸟哥还是比较习惯使用 shell script 来撰写防火墙规则，而且此一特色还可以用在呼叫其它的 scripts ， 可以让防火墙规则具有较为灵活的使用方式。好了，那就来谈谈如何设定咱们的防火墙规则吧！

规则草拟

鸟哥底下介绍的这个防火墙，其实可以用来作为路由器上的防火墙，也可以用来作为本机的防火墙。 假设硬件联机如同图二所示那样的环境， Linux 主机本身也是内部 LAN 的路由器！ 亦即是一个简单的 IP 分享器的功能啦！假设鸟哥网络接口有底下这些：

- 外部网络使用 eth1 (如果是拨接，有可能是 ppp0，请针对您的环境来设定)；
- 内部网络使用 eth0 ，且内部使用 192.168.1.0/24 这个 Class ；
- 主机预设开放的服务有 WWW, SSH, SMTP 等等；

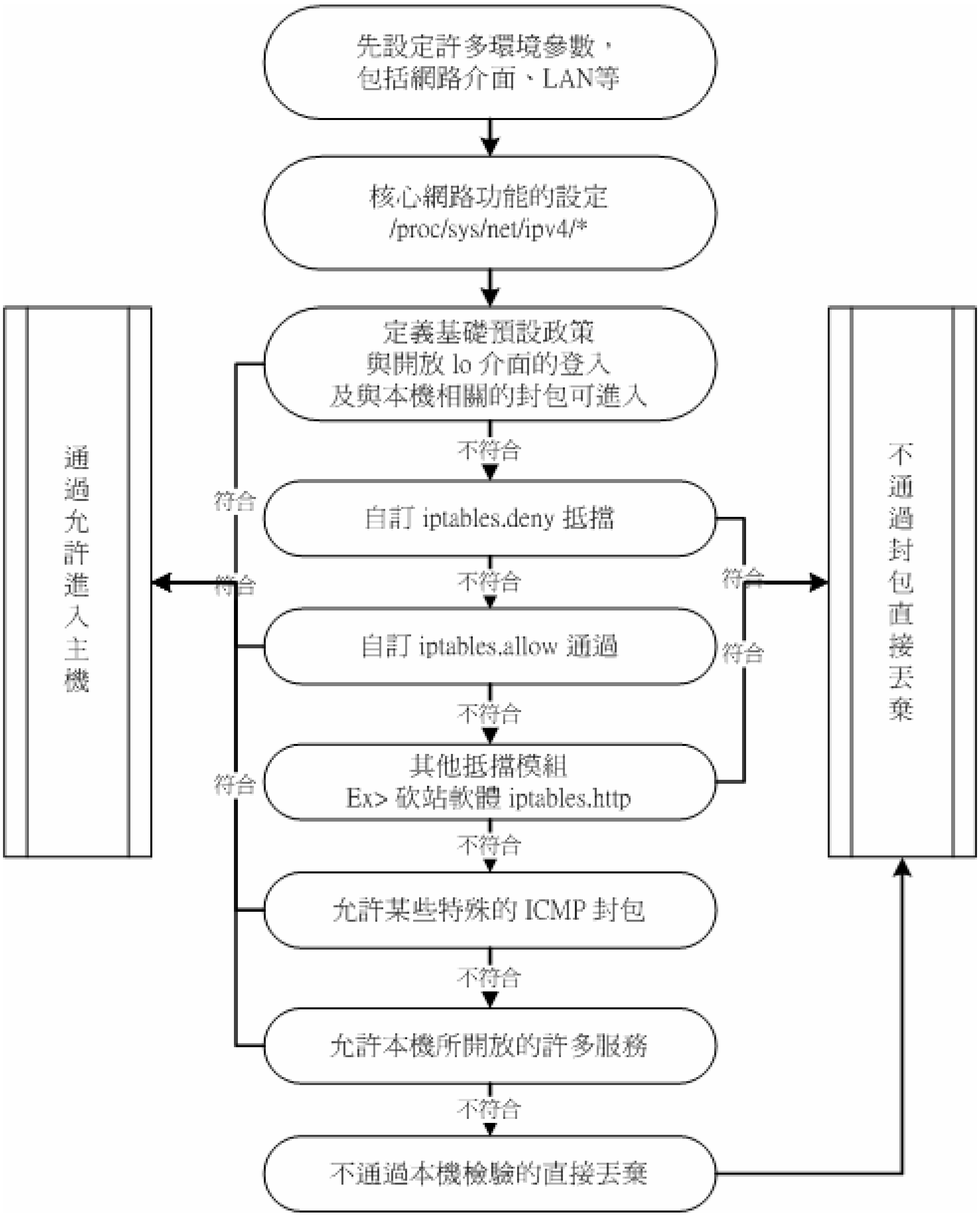
由于希望将信任网域 (LAN) 与不信任网域 (Internet) 整个分开的完整一点， 所以希望你可以在 Linux 上面安装两块以上的实体网卡，将两块网卡接在不同的网域，这样可以避免很多问题。 至于最重要的防火墙规则是：『关闭所有的联机，仅开放特定的服务』模式。 而且假设内部使用者已经受过良好的训练，因此在 filter table 的三条链个预设政策是：

- INPUT 为 DROP
- OUTPUT 及 FORWARD 为 ACCEPT

为了未来修改的方便，鸟哥将整个 script 拆成三部分，分别是：

- iptables.rule：设定最基本的规则，包括清除防火墙规则、加载模块、设定服务可接受等；
- iptables.deny：设定抵挡某些恶意主机的进入；
- iptables.allow：设定允许某些自订的后门来源主机！

鸟哥底下预计提供的防火墙流程是这样的：



图九、防火墙规则的流程

原则上，内部 LAN 主机与主机本身的开放度很高，因为 Output 与 Forward 是完全开放不理的！对于小家庭的主机是可以接受的，因为我们内部的计算机数量不多，而且人员都是熟悉的， 所以不需要特别加以控管！但是：『在大企业的内部，这样的规划是很不合格的， 因为您不能保证内部所有的人都可以按照您的规定来使用 Network ！』也就是说『家贼难防』呀！ 因此，连 Output 与 Forward 都需要特别加以管理才行！

实际设定

事实上，我们在设定防火墙的时候，不太可能会一个一个指令的输入，通常是利用 shell scripts 来帮我们达成这样的功能呐！底下是利用上面的流程图所规划出来的防火墙 scripts，您可以参考看看， 但是您需要将环境修改成适合您自己的环境才行喔！

```
[root@linux ~]# mkdir -p /usr/local/virus/iptables
[root@linux ~]# cd /usr/local/virus/iptables
[root@linux iptables]# vi iptables.rule
#!/bin/bash

# 请先输入您的相关参数，不要输入错误了！
  EXTIF="eth1"              # 这个是可以连上 Public IP 的网络接口
  INIF="eth0"               # 内部 LAN 的连接接口；若无请填 ""
  INNET="192.168.1.0/24"    # 内部 LAN 的网域，若没有内部 LAN 请设定为 ""
  export EXTIF INIF INNET

# 第一部份，针对本机的防火墙设定！###########################
# 1. 先设定好核心的网络功能：
  echo "1" > /proc/sys/net/ipv4/tcp_syncookies
  echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
  for i in /proc/sys/net/ipv4/conf/*/rp_filter; do
        echo "1" > $i
  done
  for i in /proc/sys/net/ipv4/conf/*/log_martians; do
        echo "1" > $i
  done
  for i in /proc/sys/net/ipv4/conf/*/accept_source_route; do
        echo "0" > $i
  done
  for i in /proc/sys/net/ipv4/conf/*/accept_redirects; do
        echo "0" > $i
  done
  for i in /proc/sys/net/ipv4/conf/*/send_redirects; do
        echo "0" > $i
  done

# 2. 清除规则、设定预设政策及开放 lo 与相关的设定值
  PATH=/sbin:/usr/sbin:/bin:/usr/bin; export PATH
  iptables -F
  iptables -X
  iptables -Z
  iptables -P INPUT   DROP
  iptables -P OUTPUT  ACCEPT
  iptables -P FORWARD ACCEPT
  iptables -A INPUT -i lo -j ACCEPT
  iptables -A INPUT -m state --state RELATED -j ACCEPT

# 3. 启动额外的防火墙 script 模块
```

```
  if [ -f /usr/local/virus/iptables/iptables.deny ]; then
        sh /usr/local/virus/iptables/iptables.deny
  fi
  if [ -f /usr/local/virus/iptables/iptables.allow ]; then
        sh /usr/local/virus/iptables/iptables.allow
  fi
  if [ -f /usr/local/virus/httpd-err/iptables.http ]; then
        sh /usr/local/virus/httpd-err/iptables.http
  fi
  iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT

# 4. 允许某些类型的 ICMP 封包进入
  AICMP="0 3 3/4 4 11 12 14 16 18"
  for tyicmp in $AICMP
  do
     iptables -A INPUT -i $EXTIF -p icmp --icmp-type $tyicmp -j ACCEPT
  done

# 5. 允许某些服务的进入，请依照您自己的环境开启
# iptables -A INPUT -p TCP -i $EXTIF --dport  22  -j ACCEPT   # SSH
# iptables -A INPUT -p TCP -i $EXTIF --dport  25  -j ACCEPT   # SMTP
# iptables -A INPUT -p UDP -i $EXTIF --sport  53  -j ACCEPT   # DNS
# iptables -A INPUT -p TCP -i $EXTIF --sport  53  -j ACCEPT   # DNS
# iptables -A INPUT -p TCP -i $EXTIF --dport  80  -j ACCEPT   # WWW
# iptables -A INPUT -p TCP -i $EXTIF --dport 110  -j ACCEPT   # POP3
# iptables -A INPUT -p TCP -i $EXTIF --dport 443  -j ACCEPT   # HTTPS

# 第二部份，针对后端主机的防火墙设定！###############################
# 1. 先加载一些有用的模块
  modules="ip_tables iptable_nat ip_nat_ftp ip_nat_irc ip_conntrack
ip_conntrack_ftp ip_conntrack_irc"
  for mod in $modules
  do
        testmod=`lsmod | grep "${mod} "`
        if [ "$testmod" == "" ]; then
                modprobe $mod
        fi
  done

# 2. 清除 NAT table 的规则吧！
  iptables -F -t nat
  iptables -X -t nat
  iptables -Z -t nat
```

```
  iptables -t nat -P PREROUTING  ACCEPT
  iptables -t nat -P POSTROUTING ACCEPT
  iptables -t nat -P OUTPUT      ACCEPT

# 3. 开放成为路由器，且为 IP 分享器！
  if [ "$INIF" != "" ]; then
    iptables -A INPUT -i $INIF -j ACCEPT
    echo "1" > /proc/sys/net/ipv4/ip_forward
    if [ "$INNET" != "" ]; then
      for innet in $INNET
      do
        iptables -t nat -A POSTROUTING -s $innet -o $EXTIF -j MASQUERADE
      done
    fi
  fi

# 4. 内部服务器的设定:
# iptables -t nat -A PREROUTING -p tcp -i $EXTIF --dport 80  \
         -j DNAT --to 192.168.1.210:80
```

特别留意上面程序代码的特殊字体部分，基本上，你只要修改一下最上方的接口部分， 应该就能够运作这个防火墙了。不过因为每个人的环境都不相同， 因此你在设定完成后，依旧需要测试一下才行喔！不然，出了问题不要怪我啊！.... 再来看一下关于 iptables.allow 的内容是如何？假如我要让一个 140.116.44.0/24 这个网域的所有主机来源可以进入我的主机的话，那么这个档案的内容可以写成这样：

```
[root@linux iptables]# vi iptables.allow
#!/bin/bash
# 底下则填写你允许进入本机的其它网域或主机啊！
  iptables -A INPUT -i $EXTIF -s 140.116.44.0/24 -j ACCEPT

# 底下则是关于抵挡的档案设定法！
[root@linux iptables]# vi iptables.deny
#!/bin/bash
# 底下填写的是『你要抵挡的那个咚咚！』
  iptables -A INPUT -i $EXTIF -s 140.116.44.254 -j DROP

[root@linux iptables]# chmod 700 iptables.*
```

将这三个档案的权限设定为 700 且只属于 root 的权限后，就能够直接执行 iptables.rule 啰！ 不过要注意的是，在上面的案例当中，鸟哥预设将所有的服务的通道都是关闭的！ 所以你必须要到本机防火墙的第 5 步骤处将一些批注符号 (#) 解开才行。 同样的，如果有其它更多的 port 想要开启时，一样需要增加额外的规则才行喔！

不过，还是如同前面我们所说的，这个 firewall 仅能提供基本的安全防护，其它的相关问题还需要再测试测试呢！ 此外，如果你希望一开机就自动执行这个 script 的话，请将这个档案的完整档名写入 /etc/rc.d/rc.local 当中，有点像底下这样：

```
[root@linux ~]# vi /etc/rc.d/rc.local
.....其它省略.....
# 1. Firewall
/usr/local/virus/iptables/iptables.rule
.....其它省略.....
```

上述三个档案请你不要在 Windows 系统上面编辑后传送到 Linux 上运作，因为 Windows 系统的断行字符问题， 将可能导致该档案无法执行。建议你直接到底下去下载，传送到 Linux 后可以利用 dos2unix 指令去转换断行字符！ 就不会有问题！

- http://linux.vbird.org/download/index.php?action=detail&fileid=43

这就是一个最简单、阳春的防火墙。同时，这个防火墙还可以具有最阳春的 IP 分享器的功能呢！ 也就是在 iptables.rule 这个档案当中的第二部分了。 这部分我们在下一节会再继续介绍的。

NAT 主机的设定

呼呼！终于来到这个地方了！我们准备要架设一个路由器的延伸服务器，就称之为 NAT 主机。 NAT 是什么呢？简单的说，你可以称他为内部 LAN 主机的『 IP 分享器』啦！

NAT 的全名是 Network Address Translation，字面上的意思是『网络地址的转换』。由字面上的意思我们来想一想， TCP/IP 的网络封包不是有 IP 地址吗？那 IP 地址不是有来源与目的吗？我们的 iptables 指令就能够修改 IP 封包的表头数据， 嘿嘿！连目标或来源的 IP 地址都可以修改呢！甚至连 TCP 封包表头的 port number 也能修改！真是有趣！

NAT 主机的功能可以达到类似图二所介绍的类似 IP 分享的功能之外， 还可以达到类似图四所介绍的 DMZ (非军事区) 的功能！这完全取决于我们的 NAT 是修改： (1)来源 IP 还是 (2)目标 IP ！底下我们就来聊一聊吧！ ^_^

什么是 NAT？ SNAT？ DNAT？

在谈到 NAT 的实际运作之前，让我们再来看一下比较简单的封包透过 iptables 而传送到后端主机的流程(请往前参考图八)。当网络布线如图二的架构， 若内部 LAN 有任何一部主机想要传送封包出去时，那么这个封包要如何透过 Linux 主机而传送出去？ 他是这样的：

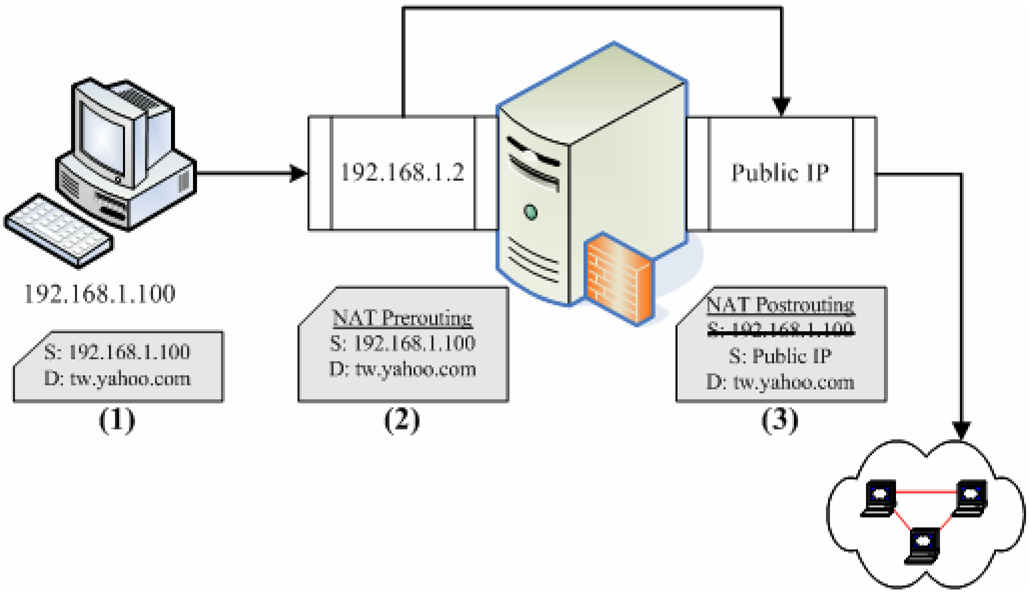
1. 先经过 NAT table 的 PREROUTING 链；
2. 经由路由判断确定这个封包是要进入本机与否，若不进入本机，则下一步；
3. 再经过 Filter table 的 FORWARD 链；
4. 通过 NAT table 的 POSTROUTING 链，最后传送出去。

NAT 主机的重点就在于上面流程的第 1,4 步骤，也就是 NAT table 的两条重要的链：PREROUTING 与 POSTROUTING。 那这两条链有什么重要的功能呢？重点在于修改 IP 嘛！但是这两条链修改的 IP 是不一样的！ POSTROUTING 在修改来源 IP ，PREROUTING 则在修改目标 IP 。 由于修改的 IP 不一样，所以就

称为 来源 NAT (Source NAT, SNAT) 及目标 NAT (Destination NAT, DNAT)。我们先来谈一谈 IP 分享器功能的 SNAT 吧!

- 来源 NAT, SNAT

你应该有听说过 IP 分享器这个玩意儿，他可以让你家庭里的好几部主机同时透过一条 ADSL 网络联机到 Internet 上面，例如图二联机的方式来说，那个 Linux 主机就是 IP 分享器啦!那么他是如何达到 IP 分享的功能？就是透过 NAT 表格的 POSTROUTING 来处理的。假设你的网络布线如图二所示， 那么 NAT 主机是如何处理这个封包的呢？

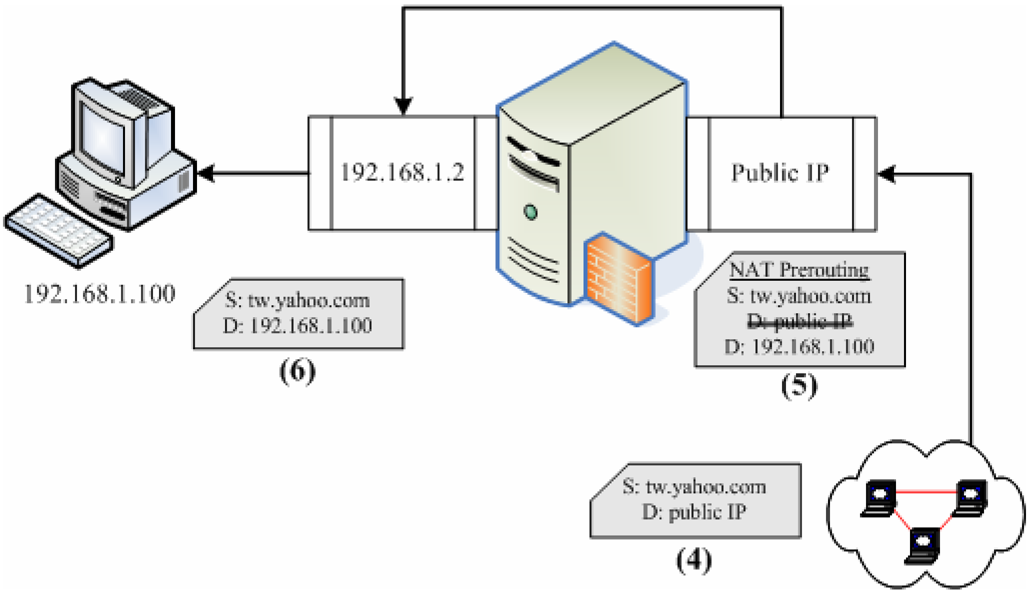


图十、SNAT 封包传送出去的示意图

如上图所示，在客户端 192.168.1.100 这部主机要联机到 http://tw.yahoo.com 去时，他的封包表头会如何变化？

1. 客户端所发出的封包表头中，来源会是 192.168.1.100 ，然后传送到 NAT 这部主机;
2. NAT 这部主机的内部接口 (192.168.1.2) 接收到这个封包后，会主动分析表头资料， 因为表头数据显示目的并非 Linux 本机，所以开始经过路由， 将此封包转到可以连接到 Internet 的 Public IP 处;
3. 由于 private IP 与 public IP 不能互通，所以 Linux 主机透过 iptables 的 NAT table 内的 Postrouting 链将封包表头的来源伪装成为 Linux 的 Public IP ，并且将两个不同来源 (192.168.1.100 及 public IP) 的封包对应写入暂存内存当中， 然后将此封包传送出去了;

此时 Internet 上面看到这个封包时，都只会知道这个封包来自那个 Public IP 而不知道其实是来自内部啦。 好了，那么如果 Internet 回传封包呢？又会怎么作？



图十一、SNAT 封包接收的示意图

4. 在 Internet 上面的主机接到这个封包时，会将响应数据传送给那个 Public IP 的主机；
5. 当 Linux NAT 主机收到来自 Internet 的响应封包后，会分析该封包的序号，并比对刚刚记录到内存当中的数据， 由于发现该封包为后端主机之前传送出去的，因此在 NAT Prerouting 链中，会将目标 IP 修改成为后端主机，亦即那部 192.168.1.100，然后发现目标已经不是本机 (public IP)， 所以开始透过路由分析封包流向；
6. 封包会传送到 192.168.1.2 这个内部接口，然后再传送到最终目标 192.168.1.100 机器上去！

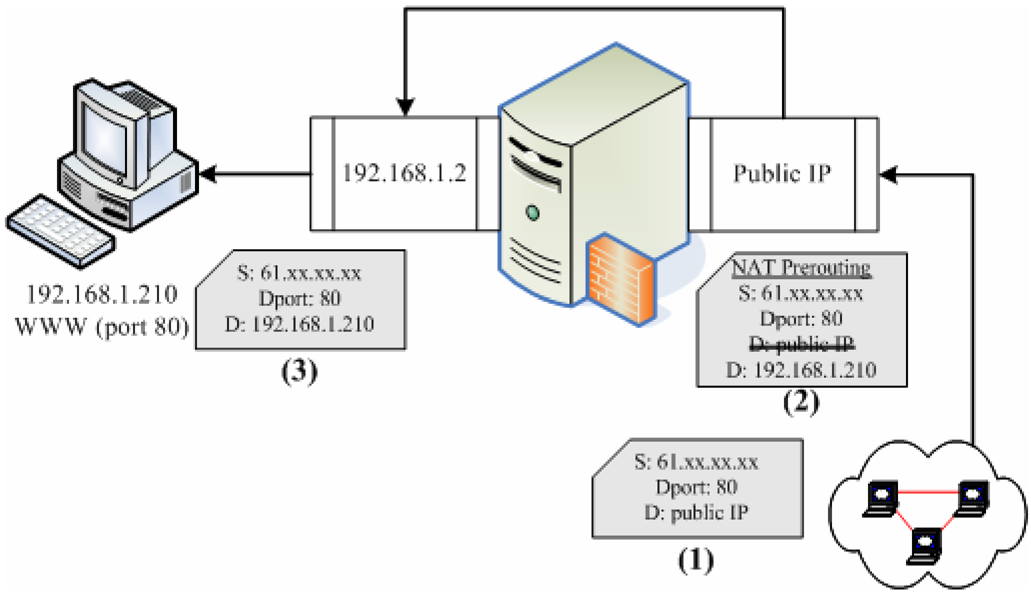
经过这个流程，您就可以发现到，所有内部 LAN 的主机都可以透过这部 NAT 主机联机出去， 而大家在 Internet 上面看到的都是同一个 IP (就是 NAT 那部主机的 public IP 啦！)， 所以，如果内部 LAN 主机没有连上不明网站的话，那么内部主机其实是具有一定程度的安全性的啦！ 因为 Internet 上的其它主机没有办法主动攻击你的 LAN 内的 PC 嘛！所以我们才会说， NAT 最简单的功能就是类似 IP 分享器啦！那也是 SNAT 的一种。

Tips:
NAT 主机与路由器有啥不同？基本上，NAT 主机一定是路由器，不过， NAT 主机由于会修改 IP 表头数据， 因此与单纯转递封包的路由器不同。最常见的 IP 分享器就是一个路由器，但是这个 IP 分享器一定会有一个 Public IP 与一个 Private IP，让 LAN 内的 Private IP 可以透过 IP 分享器的 Public IP 传送出去喔！至于路由器通常两边都是 Public IP 或同时为 Private IP。



- 目标 NAT, DNAT

SNAT 主要是应付内部 LAN 连接到 Internet 的使用方式，至于 DNAT 则主要用在内部主机想要架设可以让 Internet 存取的服务器啦！ 就有点类似图四的 DMZ 内的主机啊！底下也先来谈一谈 DNAT 的运作吧！



图十二、DNAT 的封包传送示意图

如上图十二所示，假设我的内部主机 192.168.1.210 启动了 WWW 服务，这个服务的 port 开启在 port 80 ， 那么 Internet 上面的主机 (61.xx.xx.xx) 要如何连接到我的内部服务器呢？当然啦， 还是得要透过 Linux NAT 主机嘛！所以这部 Internet 上面的机器必须要连接到我们的 NAT 的 public IP 才行。

1. 外部主机想要连接到目的端的 WWW 服务，则必须要连接到我们的 NAT 主机上头；
2. 我们的 NAT 主机已经设定好要分析出 port 80 的封包，所以当 NAT 主机接到这个封包后， 会将目标 IP 由 public IP 改成 192.168.1.210 ，且将该封包相关信息记录下来，等待内部服务器的响应；
3. 上述的封包在经过路由后，来到 private 接口处，然后透过内部的 LAN 传送到 192.168.1.210 上头！
4. 192.186.1.210 会响应数据给 61.xx.xx.xx ，这个回应当然会传送到 192.168.1.2 上头去；
5. 经过路由判断后，来到 NAT Postrouting 的链，然后透过刚刚第二步骤的记录，将来源 IP 由 192.168.1.210 改为 public IP 后，就可以传送出去了！ (类似图十的状态！)。

其实整个步骤几乎就等于 SNAT 的反向传送哩！这就是 DNAT 啰！很简单吧！

最阳春 NAT 主机： IP 分享功能

在 Linux 的 NAT 主机服务当中，最常见的就是类似图二的 IP 分享器功能了。 而由刚刚的介绍你也该知道，这个 IP 分享器的功能其实就是 SNAT 啦！作用就只是在 iptables 内的 NAT 表格当中，那个路由后的 POSTROUTING 链进行 IP 的伪装就是了。另外， 你也必须要了解，你的 NAT 主机必须要有一个 public IP 接口，以及一个内部 LAN 连接的 private IP 接口才行。

同样的，我的假设是这样的：

- 外部接口使用 eth1 ，这个接口具有 public IP 喔；
- 内部接口使用 eth0 ，假设这个 IP 为 192.168.1.2 ；

记住！当你利用前面几章谈到的数据来设定你的网络参数后，务必要进行路由的检测， 因为在 NAT 主机的设定方面，最容易出错的地方就是路由了！尤其是在拨皆产生 ppp0 这个对外接口的环境下， 这个问题最严重。反正你要记得：『如果你的 public IP 取得的方式是拨接或 cable modem 时，你的设定档 /etc/sysconfig/network, ifcfg-eth0, ifcfg-eth1 等档案，千万不要设定 GATEWAY 啦！』否则就会出现两个 default gateway ，反而会造成问题。

如果你刚刚已经下载了 iptables.rule ，那么该档案内已经含有 NAT 的脚本了！ 你可以看到该档案的第二部份关于 NAT 主机的部分，应该有看到底下这几行：

```
iptables -A INPUT -i $INIF -j ACCEPT
# 这一行在让 NAT 主机可接受来自内部 LAN 的封包
echo "1" > /proc/sys/net/ipv4/ip_forward
# 上头这一行则是在让你的 Linux 具有 router 的能力
iptables -t nat -A POSTROUTING -s $innet -o $EXTIF -j MASQUERADE
# 这一行最关键！就是加入 nat table 封包伪装！
```

重点在那个『 MASQUERADE 』！这个设定值就是『 IP 伪装成为封包出去 (-o) 的那块装置上的 IP 』！以上面的例子来说，就是 $EXTIF ，也就是 eth1 啦！ 所以封包来源只要来自 $innet (也就是内部 LAN 的其它主机) ，只要该封包可透过 eth1 传送出去，那就会自动的修改 IP 的来源表头成为 eth1 的 public IP 啦！就这么简单！ 你只要将 iptables.rule 下载后，并设定好你的内、外网络接口， 执行 iptables.rule 后，你的 Linux 就拥有主机防火墙以及 NAT 主机的功能了！

- LAN 内其它 PC 的设定

上面提到的是 NAT 主机的设定，那么在 LAN 内的其它 PC 网络参数要如何设定呢？很简单啊， 只要记得底下的参数值即可：

- NETWORK 为 192.168.1.0
- NETMASK 为 255.255.255.0
- BROADCAST 为 192.168.1.255
- IP 可以设定 192.168.1.1 ~ 192.168.1.254 间，不可重复！
- 通讯闸 (Gateway) 需要设定为 192.168.1.2 (NAT 主机的 Private IP)
- DNS (/etc/resolv.conf) 需设定为 168.95.1.1 (Hinet) 或 139.175.10.20 (Seed Net)， 这个请依您的 ISP 而定；

这样就搞定一部阳春的 NAT 主机了！简单的要命啊！
事实上，除了 IP 伪装 (MASQUERADE) 之外，我们还可以直接指定修改 IP 封包表头的来源 IP 呢！ 举例来说，如下面这个例子：

```
范例：将要由 eth1 传送出去的封包，封包来源改为 192.168.200.250
[root@linux ~]# iptables -t nat -A POSTROUTING -o eth1 \
>  -j SNAT --to 192.168.200.250

范例：同上，但封包来源为 192.168.200.210~220
[root@linux ~]# iptables -t nat -A POSTROUTING -o eth1 \
>  -j SNAT --to 192.168.200.210-192.168.200.210
```

这样也可以修改网络封包的来源 IP 资料喔！不过，除非你使用的是固定 IP ， 且有多个 IP 可以对外联机，否则一般使用 IP 伪装即可，不需要使用到这个 SNAT 吧？ 当然，你也可能有自己的独特的环境啦！ ^_^

---

iptables 的额外核心模块功能

如果你刚刚在 iptables.rule 内的第二部分有仔细看的话， 那有没有觉得很奇怪，为何我们需要加载一些有用的模块？举例来说， ip_nat_ftp 及 ip_net_irc ？ 这是因为很多通讯协议使用的封包传输比较特殊，尤其是 FTP 档案传输使用到两个 port 来处理资料！ 这个部分我们会在 FTP 章节再次的详谈，在这里你要先知道，我们的 iptables 提供很多好用的模块， 这些模块可以辅助封包的过滤用途，让我们可以节省很多 iptables 的规则拟定， 好棒的呐！ ^_^

在防火墙后端之网络服务器 DNAT 设定

既然可以做 SNAT 的 IP 分享功能，我们当然可以使用 iptables 做出 DMZ 啦！ 但是再次重申，不同的服务器封包传输的方式可能有点差异，因此，建议新手不要玩这个咚咚！ 否则很容易导致某些服务无法顺利对 Internet 提供的问题。

先来谈一谈，如果我想要处理 DNAT 的功能时， iptables 要如何下达指令？ 另外，你必须要知道的是，DNAT 用到的是 nat table 的 Prerouting 链喔！不要搞错了。

```
范例：将连接到 eth1 接口的 port 80 传导到内部的 192.168.1.210
[root@linux ~]# iptables -t nat -A PREROUTING -p tcp -i eth1 \
> --dport 80 -j DNAT --to 192.168.1.210:80
```

那个『 -j DNAT --to IP[:port] 』就是精髓啦！代表从 eth1 这个接口传入的，且想要使用 port 80 的服务时， 将该封包重新传导到 192.168.1.210:80 的 IP 及 port 上面！可以同时修改 IP 与 port 呢！真方便。 其它还有一些较进阶的 iptables 使用方式，如下所示：

```
-j REDIRECT --to-ports <port number>
# 这个也挺常见的，基本上，就是进行本机上面 port 的转换就是了！
# 不过，特别留意的是，这个动作仅能够在 nat table 的 PREROUTING 以及
# OUTPUT 链上面实行而已喔！

范例：将要求与 80 联机的封包转递到 8080 这个 port
[root@linux ~]# iptables -t nat -A PREROUTING -p tcp  --dport 80 \
> -j REDIRECT --to-ports 8080
# 这玩意最容易在您使用了非正规的 port 来进行某些 well known 的协议，
# 例如使用 8080 这个 port 来启动 WWW ，但是别人都以 port 80 来联机，
# 所以，您就可以使用上面的方式来将对方对您主机的联机传递到 8080 啰！
```

至于更多的用途，那就有待你自己的发掘啰！ ^_^

重点回顾

- 要拥有一部安全的主机，必须要有良好的主机权限设定；随时的更新套件；定期的重要数据备份；完善的员工教育训练。 仅有防火墙是不足够的；
- 防火墙最大的功能就是帮助你『限制某些服务的存取来源』，可以管制来源与目标的 IP ；
- 防火墙依据封包抵挡的阶层，可以分为 Proxy 以及 IP Filter (封包过滤) 两种类型；
- 为了将整个网络的信任 (LAN) 与不信任 (Internet) 网域完整切割，防火墙通常具有两个实体网络接口， 分别连结信任与不信任网域；
- 在防火墙内，但不在 LAN 内的服务器所在网域，通常被称为 DMZ (非军事区)，如图四所示；
- 封包过滤机制的防火墙，通常至少可以分析 IP, port, flag (如 TCP 封包的 SYN), MAC 等等；
- 防火墙对于病毒的抵挡并不敏感；
- 防火墙对于来自内部的网络误用或滥用的抵挡性可能较不足；
- 并不是架设防火墙之后，系统就一定很安全！还是需要更新套件漏洞以及管制使用者及权限设定等；
- 核心 2.4 以后的 Linux 使用 iptables 作为防火墙的软件；
- 防火墙的订定与『规则顺序』有很大的关系；若规则顺序错误，可能会导致防火墙的失效；
- iptables 的预设 table 共有三个，分别是 filter, nat 及 mangle ，惯用者为 filter (本机) 与 nat (后端主机)。
- filter table 主要为针对本机的防火墙设定，依据封包流向又分为 INPUT, OUTPUT, FORWARD 三条链；
- nat table 主要针对防火墙的后端主机，依据封包流向又分为 PREROUTING, OUTPUT, POSTROUTING 三条链， 其中 PREROUTING 与 DNAT 有关， POSTROUTING 则与 SNAT 有关；
- iptables 的防火墙为规则比对，但所有规则都不符合时，则以预设政策 (policy) 作为封包的行为依据；
- 核心本身有提供很多网络相关功能，针对 IPv4 之设定值都在 /proc/sys/net/ipv4/* 内；
- iptables 的指令列当中，可以下达的参数相当的多，当下达 -j LOG 的参数时，则该封包的流程会被纪录到 /var/log/messages 当中；
- 防火墙可以多重设定，例如虽然已经设定了 iptables ，但是仍然可以持续设定 TCP Wrappers ，因为谁也不晓得什么时候 iptables 会有漏洞～或者是规则规划不良！

课后练习

- 为什么我架设了防火墙，我的主机还是可能中毒？

  防火墙不是万灵丹，他还是可能被病毒或者是木马程序所入侵的！ 此外，如果您的主机本身就已经提供了多个网络服务，则当该网络服务的套件有漏洞时，防火墙仍然无法克服该服务的漏洞的！因此仍然需要持续的进行主机的监视工作

- 请说明为何架设了防火墙，我的主机还是可能被入侵？入侵的依据可能是什么方法？

  因为防火墙仅是抵挡某些不受欢迎的封包，如果您有开放 WWW 的服务时，则要求您主机 port 80 的封包将可直接进入您的主机，万一 WWW 套件有漏洞时，那么就可能被入侵了！所以套件的更新很重要！

- 我们知道核心为 2.4 的 Linux 使用的防火墙机制为 iptables ，请问，如何知道我的 Linux 核心版本？

  利用 uname -r 可以查得！

- 请列出 iptables 预设的两个主要的 table ，以及各个 table 里面的 chains 与各个 chains 所代表的意义；

  filter 为预设的 Table，里头预设的链有：

  - INPUT：为来自外部，想要进入主机的封包；
  - OUTPUT：为来自主机，想要离开主机的封包；
  - FORWARD：为主机内部网域与外部网域的封包(不论进或者出)，但该封包不会进入主机。

  还有 nat 这个 table：

  - PREROUTING：进行路由之前的封包传送过程
  - OUTPUT：离开主机的封包传送过程；
  - POSTROUTING：已经经过路由了，然后才进行的过滤规则。

- 什么是 iptables 的预设政策 (Policy)？若我要针对 filter 的 INPUT 做成 DROP 的预设政策，指令如何下达？

  当封包的所有属性都不在防火墙的规则当中时，那么这个封包能否顺利的通过防火墙，则以 Policy 作为这个封包的最终动作了！
  iptables -P INPUT DROP

- 假设今天我的 Linux 仅是作为 Client 之用，并没有对 Internet 进行任何服务， 那么您的防火墙规划应该如何设定比较好？！

  既然没有对 Internet 提供任何服务，那么(1)请将所有的对外埠口先关闭吧！(2)防火墙规则当中，最重要的是 INPUT 的 Policy 一定要 DROP ，然后将『 iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT 』即可！

- 我要将来自 192.168.1.50 这个 IP 来源的封包，只要是向我的 21~23 埠口要求的封包，就将他抵挡，应该如何下达 iptables 指令？

  iptables -A INPUT -p tcp -s 192.168.1.50 --dport 21:23 -j DROP

- 我要将我自己主机 ping 的响应功能取消，应该如何下达 iptables 的指令？

  因为 ping 能否响应用的是 icmp 的 type 8 (请参考网络基础内的 ICMP 相关内容)，所以我可以这样做：
  iptables -I INPUT -p icmp --icmp-type 8 -j DROP

- 请说明为何这个指令是错误的？『iptables -A INPUT -p udp --syn -s 192.168.0.20 -j DROP』？

因为只有 TCP 封包才会具有 SYN 的标志， UDP 并没有 SYN 的标志啊！所以上面的指令是错误的

- DNS 的要求是必须的，那么我该如何设定我的主机可以接受要求 DNS 的响应呢？

  因为 DNS 的来源是 port 53 ，因此要接受来自 port 53 的封包就成为了：
  iptables -A INPUT -p udp --sport 53 -j ACCEPT
  iptables -A INPUT -p tcp --sport 53 -j ACCEPT

- 如何取消 iptables 在我的系统上面？

  先要清除规则后，才能够将 iptables 移除！不过，我们主要将规则清除即可！
  iptables -F; iptables -X; iptables -Z
  iptables -t nat -F; iptables -t nat -X; iptables -t nat -Z

- 如何储存目前的防火墙机制，以及如何将上次储存下来的机制回复到目前的系统中？

  请利用 iptables-save 以及 iptables-restore 这两个指令，配合命令重导向即可！

- 如果你的区网当中有个 PC 使用者老是连上 Internet 乱搞，你想要将他的 IP 锁住，但他总是有办法修改成其它 IP 来连外， 那你该怎么办？让他无法继续连外？

  可以利用封锁网络卡卡号 MAC 来处理！

参考数据

中文网站：

- http://www.study-area.org/linux/servers/linux_nat.htm
- http://linux.tnc.edu.tw/techdoc/firewall/
- http://www.linuxyes.com/tw/tutorial/iptables.html

英文网站：

- http://www.netfilter.org/
- http://www.linuxguruz.org/iptables/
- http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html
- http://www.interhack.net/pubs/fwfaq/

其它书籍与数据：

- Robert L. Ziegler 着，朱亮恺等译，『实战 Linux 防火墙--iptables 应用全搜录』，上奇出版社，2004。
- 本机的核心文件：/usr/src/linux-{version}/networking/ip-sysctl.txt

- iptables 的内建 tables 与各个 chain 的相关性：
  http://ebtables.sourceforge.net/br_fw_ia/bridge3b.png
- 核心参数的相关说明：
  http://www.study-area.org/tips/adv-route/Adv-Routing-HOWTO-12.html

---

申请合法的主机名称

最近更新日期：2006/09/16

呵呵！在您读完了网络基础，设定好了防火墙，解决了认识埠口的问题，并且架设了个人简易的防火墙之后， 总算是准备要开始来给他进入 Server 的架设了！服务器架设的步骤里面，很重要的一点是『您的主机名称必须要在 Internet 上面可以被查询』才好！由网络基础我们知道网络上的设备(主机或其它网络媒体)仅认识 IP ，但人类对于 IP 记忆力又不佳，所以才会以主机名称来取代 IP。不过， 您的主机名称要能够被查询到才有用啊！这个时候，一个『合法』的主机名称就很重要了！那要合法的主机，就得要让 DNS 系统能够找的到您的主机啊！不过，如果我们的主机是使用拨接得到的不固定 IP 呢？又该如何申请 DNS 主机名称？那就得要使用动态 DNS 的系统啰！在这个章节中，我们主要在介绍 Client 端的设定，而不是在设定 DNS 主机喔！ ^_^

本章的行前准备工作

在这个章节当中，我们将会介绍如何申请一个合法的主机名称。目前 Internet 上面使用的主机名称都是透过所谓的 DNS 系统，而你想要取得一个 DNS 的主机名称，就必须要『注册』，所谓的『注册』就是要钱去申请啦！ 当然也有免费提供主机名称的服务啦！在这个章节当中鸟哥不会介绍如何架设一部 DNS 主机， 而是介绍如何利用注册或免费申请的方式来达成主机名称的取得。在这一章开始前，您最好先复习一下：

- 因为需要安装软件，你需要知道原始码、tarball 及安装的流程；
- 取得主机名称你需要知道网络基础章节内的 DNS 简介；
- 由于谈到很多 IP 的观念，你必须要知道网络基础章节内的自动取得 IP 与固定 IP 的意义；

如果没问题的话，那就赶紧来玩玩！

为何需要主机名称

如果你已经将网络基础那一章看完的话，应该会知道其实我们的 TCP/IP 环境只要有 IP 与正确的路由即可联机了。 那么你申请主机名称要干嘛？因为『没办法啊！人脑太不中用了！』 举例来说，你可以背出来我们常上去查数据的 www.google.com 的 IP 吗？报告！鸟哥没办法背出来～

因为 IP 是那么难背的东西，而且，如果您的 IP 又是类似拨接的不固定的 IP 时，那还更伤脑筋呢！ 因此我们才会习惯以熟悉的英文字符串来做为主机名称， 然后让『这个主机名称与 IP 达成对应』，那直接记忆主机名称就行了，反正 IP 的查询就交给计算机主机来做即可！ 在这样的想法下，我们当然就需要有主机名称啦！底下咱们就来谈一谈先！

主机名称的由来

从上面的说明我们知道因为 IP 不好记，所以人类习惯使用主机名称来记忆与连上某部主机。 好在早期连上网络的计算机数量不多，所以在网络上的人们就想出一个简单的办法来进行主机名称与 IP 的对应，那就是『在每部计算机的 /etc/hosts 里面设定好主机名称与 IP 的对应表』。那么未来人们就可以直接藉由主机名称来连接上某些网络上的主机啰！

然而因为科技的发达，连上 Internet 的人们越来越多，使用 /etc/hosts 的方法已经搞不定了(只要一部计算机上线，全部的 Internet 上面的所有计算机都要重新改写 /etc/hosts！不太好吧！），这个时候领域名称系统 (Domain Name System, DNS) 就适时的出现了！

DNS 利用类似树状目录的型态，将主机名称的管理分配在不同层级的 DNS 主机当中，经由分层管理，所以每一部主机的记忆的信息就不会很多，而且异动上面也相当的容易修改！ 那么这个 DNS 的功能您知道了吗？对啦！就是『将计算机主机的名称转译成 IP 』就是了！当然啰，他的额外功能还很多，关于 DNS 的详细的解析部分我们将在后续的 DNS 主机架设章节当中在持续的加强内容，总之， 他的最大功能就是『让有意义的，人类较容易记忆的主机名称(英文字母) ，转译成为计算机所熟悉的 IP 地址！』

重点在合法授权

很多朋友都认为：『因为我想要架站，所以主机需要有个主机名称， 因此我就得要架设 DNS？』是这样吗？当然不是啰！DNS 是个很庞大的架构， 而且是串连在全球的网络当中，除非你经由『注册』的手续来让 DNS 系统承认你的主机名称的存在， 否则你架设的 DNS 只能说是一个『地下练习的测试站』而已啦！并没有用途的。

那我要如何加入 DNS 系统呢？很简单啦！首先你必须要选择一个注册单位， 并且检查出你想要注册的主机名称是否存在？主机名称是有意义的，并不是你可以随便注册的喔！ 举例来说，在台湾常见的个人网站注册主机名称为： *.idv.tw ，而公司行号则可能注册为 *.com.tw 了！这个得要特别留意。至于台湾地区的注册单位很多，你可以选择例如 Hinet 或 Seednet 之类的 ISP 来注册。当然，你也可以选择免费的 no-ip.org 来注册的。

但要请您特别注意的是，并不是所有的注册单位都提供单纯主机名称的对应功能， 所以要注册前，请『货比三家』啊！鸟哥所申请的单位分别是国外的 .org 及台湾的 Hinet 两家， Hinet 有提供 .idv.tw 的主机名称对应，还不错。 当然你也可以使用免费的 no-ip.org 来进行主机名称的注册！

Tips:
在这个章节当中，鸟哥的讲解比较少，因为很多数据都与 DNS 服务器篇 有重复，在这个章节当中鸟哥主要在介绍动态 IP 架站的一个简单主机名称申请方式啦！ ^_^



申请静态还是动态 DNS 主机名称

由上面的说明当中，我们可以很清楚的知道 DNS 系统最大的功能就是在主机名称对应 IP 的转译上面。当然啦，预设的 DNS 转译是用在『固定 IP 对应主机名称』 的方法上面的！夭寿喔！我们的小站很多都是以非固定 IP(很多人也称为浮动式 IP 、动态 IP 等等的名称) 来上网的，更有甚者， Hinet 的 ADSL 拨接都是 24 小时强制断线一次的，所以我们都得需要重新拨接上网， 而每次拨接成功后取得的 IP 可不见得相同啊，如此一来 IP 不是一直在变吗？ 那么我不就需要一直跟我上层 DNS 主机的管理员申请『变更 IP 』吗？会不会太麻烦了点？

是很麻烦啊！所以现在为了解决这个问题，很多 ISP 提供了所谓的 动态 DNS 主机服务 的功能，他是这样做的：

1. Client 端(就是您啦)每次开机或者是重新拨接，并取得一个新的 IP 之后，会向 DNS Server 端提出要求，希望 Server 端变更主机名称与 IP 的对应 (这个步骤在每个主要的 ISP 都有提供适当的 program 来提供给 client 使用)；

2. Server 端接受 Client 端的要求之后，会先去查询 Client 提供的账号密码是否正确， 正确之后就会立即修改 Server 本身对于您的主机名称的设定值。

所以啰，每次我们取得了新的 IP 之后，我们的主机名称对应的 IP 也会跟着在 DNS 系统上面更新， 如此一来，只要别人知道您的主机名称，不论您的 IP 为何，他一定可以连上您的主机 (因为 IP 跟着您的主机而变！) 这对于我们这种使用动态 IP 的人是很有帮助的！ (阿！真是造福我们这些穷苦人家的孩子呀！)

不过，还是需要注意的是，目前的主机名称申请很多是『需要钱的』！如果您需要比较稳定的主机名称对应 IP 的服务，那么花点钱来注册还是必须的，不过，如果是实验性质的网站，那么也是可以申请免费的动态 DNS 系统喔！

注册一个合法的主机名称

静态 DNS 主机注册：
好了,既然知道了 DNS 的用途,那么自然我们就需要来申请 DNS 啦!不然怎么架设网站呢？目前的静态 IP 对应主机名称的注册网站实在太多了，我们仅提出几个出来分享就是了！

- 台湾网络信息中心：http://www.twnic.net
- 国外的领域名称系统：http://www.netsol.com/
- 国外的领域名称系统：http://www.dotster.com/

动态 DNS 主机注册：

那么有没有免费的 DNS 系统呀！呵呵！当然有啰！我们要感谢造福我们这些穷苦人家的孩子的大好人～ 您可以在底下找到相关的信息：

- 国外的免费 DNS 系统：http://www.no-ip.com

静态 DNS 注册：以 Hinet 为例

静态 DNS 的申请方式其实都差不多，都是需要：

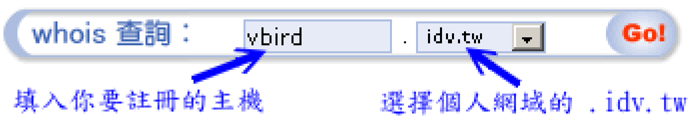
1. 先查询所想要注册的网域是否存在；
2. 进入 ISP 去申请注册您所想要的主机名称；
3. 缴费，并等待主机名称被启用。

我们以台湾蛮常见的 Hinet 这个 ISP 提供的『个人网域, .idv.tw』注册方式来说明：

---

1. 登入主画面，并查询欲注册网域是否存在

直接连结到底下的网页去：http://nweb.hinet.net/ ，并在 whois 的画面当中(右上角)选择您想要注册的主机名称，按下『Go』开始搜寻。



图一、利用 whois 查询欲注册网域是否存在

---

2. 逐步进行注册

如果确认您的主机名称没有被注册掉，那么您就可以开始注册了！同样的在上面的网站连结当中，选择『个人网域名称』就可以开始申请了！请依序一步一步办理！这里不再说明了，反正都是中文， 看的懂得啦！ ^_^



图二、个人网域逐步注册的流程示意图

---

3. 填写主机名称对应的 IP

缴费完毕之后，我们就可以开始进行主机名称的填写了！在图二的图示中按下『DNS 指定/异动』的项目， 并填入您的主机名称与密码，然后就会出现如下的画面了：



图三、主机名称与 IP 对应的填写范例

特别的给他留意，因为我们没有要架设 DNS 主机，所以当然最上方要选择『主机』的项目， 然后您可以填入三部主机名称喔！当然，这三部主机名称可以通通指向同一个 IP ，也可以不同！ 随您的便呐！需要注意到的是，您的主机名称应该是『othername.yourhost.idv.tw』后面的 yourhost.idv.tw 是不变的，前面的 othername 则可以自由选取呢！例如鸟哥上面的设定，后面均是 vbird.idv.tw ，而前面的名称就可以让我自由选择啦！

---

4. 等待 DNS 启用

在上图三当中按下『填写完请按这里』后，就等着启用吧！不过设定成功到可以使用， 其实需要一定的时间的，以鸟哥为例，第一次申请之后，大约过了 20 小时该设定才正确的启动呢！请以请耐心等候啊！不要太着急啰！ ^_^

各家的领域名称注册流程都差不多，不过，金额是有点差异的，当然，服务也就有不同啊！鸟哥的 vbird.org 领域名称则是在 http://www.godaddy.com 注册的喔！如果您不想要使用 .idv.tw 来注册的话，那么国外的 ISP 提供的 DNS 也可以考虑看看说！

动态 DNS 注册：以 no-ip 为例

如果你跟鸟哥一样使用 ADSL 拨接的方式来上网，这表示你的 IP 应该是不固定的！果真如此的话， 那想要架站就比较麻烦一点！因为上面利用 Hinet 注册的方式通常是给固定 IP 使用的， 你应该不会想要天天上去更新你的 IP 吧？此时这个 no-ip.com 所提供的免费动态 IP 对应主机名称的服务就很重要啦！我们先来申请一个主机名称来玩玩吧！ ^_^

---

1. 登入主网页，并且注册一个新账号

你必须要连上 http://www.no-ip.com 这个网站， 然后在出现的画面当中选择『 Sign-up Now 』那个项目：

图四、 no-ip 网站的注册流程之一

---

2. 开始填写识别数据

因为 no-ip 会发给您一份密码，所以在出现的如下画面中，您必须要填写『 一个可以收到邮件的合法 Email 』，以及您的身份确认数据， 这很重要，因为后续的数据都是使用您注册时的这个咚咚呐!然后再按下最底下的『SIGN UP NOW』即可！如果没有 Email 怎么办？现在免费的 email 这么多，随便申请一个吧！ ^_^

图五、 no-ip 网站的注册流程之二

---

3. 启用账号

在你申请注册一个新账号后， no-ip 会发一封信给你，信件的内容有点像底下那样：

图六、 no-ip 网站的注册流程之三

你必须要按下上图第一个箭头所指的连结后，你的账号才会正式的被启用的！ 而上图第二个连结则是在告知你可以到哪里去下载动态 DNS 的客户端软件喔！ 也就是说， no-ip 也有提供一个好用的软件给 client 端，让使用者可以『自动更新主机名称与 IP 的对应』， 呵呵！很棒吧！

---

4. 登入 no-ip 且设定主机名称与 IP 的对应

让我们回到图四的地方察看一下，不是有可以输入账号与密码的地方吗？ 请你填入你刚刚注册时

所填写的 email 地址以及密码后，就能够登入你的 no-ip 账号了。 在登入后的第一个画面左手边有点类似底下的图示：

图七、 no-ip 网站的注册流程之四

在你按下那个『add』后，画面就会产生如下的变化：

图八、 no-ip 网站的注册流程之五

请依序填写你想要的主机名称、网域名称 (通常鸟哥都建议使用 no-ip.org 这一个领域名称！)，还有你的 IP 后，咦！往下一看，竟然还有 MX 的纪录！这东西很重要！ 是用在做 mail server 时所需要的一项参数！你可以直接填写与你的完整主机名称相同的名字即可！ 填写完毕后，就按下『Create Host』吧！如果该主机名称有被使用掉的话， 屏幕会出现警告讯息，此时请再选填另外的主机名称吧！ ^_^

如果未来你想要修改或者是删除该主机名称时，可以按下图七内的 Manage 项目， 就能够进行删除与修订的工作啰！就这么简单呐！

---

5. 设定自动更新主机名称与 IP 的对应

如果系统重新开机，或者是重新拨接取得一个新的 IP 后，我们都要登入 no-ip 来修改的话， 那就太没有效率了！所以 no-ip 提供一个好用的客户端程序给使用者使用，就是在 图六 email 内的那个下载连结！你可以点选该下载连结， 在出现的窗口当中会有三种程序，包括『 Windows, Mac, Linux/BSD/Unix 』，我们当然是选择 Linux 那个项目啊！请自行下载并且将该程序移动到 Linux 主机上吧！

整个安装与启用的流程式这样的：

```
1. 编译与安装：
[root@linux ~]# cd /usr/local/src
[root@linux src]# tar -zxvf /root/noip-duc-linux.tar.gz
# 假设你将程序放置到 /root 底下时

[root@linux src]# cd noip-2.1.3
# 注意一下，这个目录里面有个文件名为 README.FIRST 的档案，务必察看一下内容！
[root@linux noip-2.1.3]# make
```

```
[root@linux noip-2.1.3]# make install
# 这样会将主程序安装在 /usr/local/bin/noip2 而主参数档放在
# /usr/local/etc/no-ip2.conf 当中！然后你必须要开始回答一些问题：

Please select the Internet interface from this list.

By typing the number associated with it.
0       eth0
1       eth1
1      <==因为鸟哥的主机对外使用 eth1 接口

Please enter the login/email string for no-ip.com  kiki@gmail.com
Please enter the password for user 'kiki@gmail.com' ***
# 上面这两个是你刚刚注册时所填写的 email 与密码喔！

Only one host [tsai.no-ip.org] is registered to this account.
It will be used.
Do you wish to run something at successful update?[N] (y/N) n

mv /tmp/no-ip2.conf /usr/local/etc/no-ip2.conf
# 重点在此！刚刚你做的设定档被放到上面这个档案中了！
```

这样就将你的 no-ip 制作完毕，而且也可以开始来执行啰！执行的方法也是很简单啦！

```
2. noip2 的程序使用：
[root@linux ~]# /usr/local/bin/noip2
# 不要怀疑！这样输入后，你在 no-ip 上面注册的主机名称，
# 就开始可以自动的产生对应了！就这么简单！

[root@linux ~]# noip2 [-CS]
参数：
-C ：重新设定参数，亦即设定刚刚我们上面输入粗体字的咚咚！
     如果您有两个以上的 no-ip 主机名称时，就一定需要使用 noip2 -C
     来重新设定参数档案！
-S ：将目前的 noip2 的状况显示出来！

[root@linux ~]# noip2 -S
1 noip2 process active.

Process 4998, started as /usr/local/src/noip-2.1.3/noip2
Using configuration from /usr/local/etc/no-ip2.conf
Last IP Address set 61.xxx.111.ddd
Account kiki@gmail.com
```

```
configured for:
        host  tsai.no-ip.org
Address check every 1 minute, directly connected via /dev/eth1.
```

嘿嘿！这样就成功了！而且每分钟 noip2 可以自动的去主网站上面进行更新呢！ 真是很不错！那如果想要一开机就启动 noip2 呢？这样做即可：

```
3. 设定开机启动：
[root@linux ~]# vi /etc/rc.d/rc.local
# 加入底下这一行：
/usr/local/bin/noip2
```

课后练习

- 请简易说明 /etc/hosts 的用途；

  这个档案是早期用在进行主机名称与 IP 的解析的，目前比较常用在内部网域的名称解析上，可以加快内部网域的反查喔！

- 请说明『合法授权』的主机名称需要做什么？

  如果想要合法授权，就需要向上层 DNS 主机『注册』才行！而且还要上层 DNS 主机管理员愿意将领域名称的解析权限授权给您啊！

- 什么是动态 DNS 系统？(仅说明 client 端)

  因为我们的 Client 拨接时，得到的 IP 都不是固定的，所以无法以 DNS 系统进行固定 IP 对应主机名称的工作！ 此时就需要动态 DNS 系统了！以 DNS 主机提供的动态更新主机名称对应 IP 的机制，可以让我们的不同 IP 对应到同一个主机名称呐！

- 如果您使用 adsl 拨接来上网设定服务器，那么该申请哪一类型的主机名称？为什么？

  因为我是以 ADSL 上网拨接，所以 IP 是不固定的，此时需要申请动态 DNS 主机的主机名称，例如 no-ip.org 等等！

参考数据

- 台湾网络信息中心：http://www.twnic.net/
- 国外的领域名称系统：http://www.netsol.com/
- 国外的领域名称系统：http://www.dotster.com/
- 国外的免费 DNS 系统：http://www.no-ip.com

远程联机服务器 Telnet / SSH / VNC / XDMCP / RSH

最近更新日期：2006/09/19

一部连上 Internet 上面的您的个人主机，最重要的是什么呢？ 大概就是如何让您自己可以联机进入自己的主机，并且进行所谓的『远程操控』了吧！ 也就是说，您可以在任何具有连上 Internet 的计算机中，以远程联机软件连上 Internet ，并藉由您主机上面的远程联机服务器软件提供的功能，直接登入您的主机来进行操控的工作！此时，您将发现 Linux 有趣又好玩的地方啰！在早期的 Unix Like 机器当中，几乎都提供 Telnet 这个远程联机服务器软件，不过， Telnet 本身是以『明码』来传送您操作的数据，安全上面是值得来思考要不要开放呐！ 这个时候就有需要了解一下传送过程中以加密动作来传送数据封包的 SSH 这个远程联机服务器软件啦！ 另外，除了纯文字接口登入主机来进行操控之外，在现在的 Linux distributions 当中， 还可以利用 X 相关的服务来帮助我们以图形接口登入喔！很棒吧！ ^_^

本章的行前准备工作

在这个章节当中我们会使用客户端的联机软件联机到主机端来操作主机，所以你必须要了解到你的主机防火墙必须要开放，并且要取消 SELinux 才行！另外，登入时会分析到的 PAM 模块也需要进行了解呐！本章后半部会介绍 X Window 的远程登入，所以你也必须对于 X Server/client 的架构有点了解才行。

- 了解网络基础，尤其网络是双向的；
- 认识网络安全当中的取消 SELinux ，以及防火墙的基本概念；
- 了解使用者与账号的相关概念；
- 认识 X Window System；
- 由于很多远程联机服务器软件系统预设并不安装，因此你必须要了解 RPM 及 yum 的使用。

远程联机服务器

远程联机服务器对我们来说，可是一项很有用的工具啊！他可以让我们更方便的管理主机。不过，方便是方便，安全性其实不很好的～所以，才要特别强调一下这个玩意儿啊！

什么是远程联机服务器

首先，我们要先来了解一下，什么是『远程联机服务器』？ 这个东西的功能为何？我想，您应该已经听过，一个良好的网络环境当中，一部开放到 Internet 上面的服务器，基本上，他可以不需要屏幕、键盘、鼠标等等的配备，只要有基本的主机板、CPU、RAM、硬盘再加上一块好一点的网络卡，并且连上 Internet ！哈哈！那么您要操控这部主机的时候，只要透过网络联机进来，然后进行任何修改即可！ 嘿！所以啰，这个时候主机自然不需要接口设备啦！

以鸟哥个人为例，目前鸟哥管理大约七、八部左右的 Unix-Like 主机， 这些主机都不在同一个地方，分布在南台湾各处！那么当有新的套件的漏洞被发布， 或者是需要进行一些额外的设定的时候，是否鸟哥本人一定要到现场吗？当然不需要， 只要透过网络联机到该主机上面，就可以进行任何工作了！真的就好像在主机前面工作一般的轻松愉快！ ^_^！这就是远程联机服务器啦！

远程联机服务器的功能当然还不只如此！举个例子来说：当您的工作需要使用到 Linux 的强大的编译功能时，那么您一定需要 Linux 对吧！而且最好是运算速度快一点的主机， 这个时候您可以将您研究室最快的那一部主机开放出来，设定一下远程联机服务器，让您的学生啦， 或者是研究室的同仁啦，可以透过这部机器帮他们进行研究的工作，这个时候，您的主机就可以让多人进行分享 Linux 运算的功能啦！

在早期的网络世界里，由于只有 Unix 机器，而且个人计算机还不流行的时候， 想要使用大型主机来进行数值程序的运算时(在我们工程界，比较常使用 Fortran 这一类的程序语言，至于 C 语言则较少碰～)，就需要向学校单位申请 Unix 工作站的账号， 并且以远程联机程序连进主机，以使用 Unix 的资源来进行我们的数值模式运算！所以啦，那个远程联机服务器的设定， 对于系统管理员是很重要的！尤其对于大型工作站类型的 Unix-Like 主机，由于很多人都需要使用到他的运算功能，或者是他的编译程序(compiler)来进行运算，这时的远程联机就更形重要啦！

那么是否每一部连上 Internet 上面的主机都应该要开放远程联机的功能呢？其实并不尽然， 还是需要针对您的主机来进行规划的，我们底下分服务器与工作站来说明：

- 服务器类型(Server)的联机程序：

在一般开放因特网服务的服务器中，由于开放的服务可能会有较为重要的信息， 而远程联机程序连进主机之后，可以进行的工作又太多了(几乎就像在主机前面工作一般！)， 因此因特网的远程联机程序通常仅针对少部分系统维护者开放而已！ 除非必要，否则 Server 类型的主机还真的不建议开放联机的服务呢！ 以鸟哥为例，我的主机提供了我们研究室使用 Mail 与 Internet 上面的 WWW 服务，如果还主动提供远程联机的话， 那么万一不小心被入侵，那可就伤脑筋了！因此，鸟哥仅开放『很小部分的网域』让系统管理员连进来， 其它来源的 IP 一律抵挡！不许使用远程联机的功能呢！

- 工作站类型(Workstation)的联机程序：

至于工作站的情况就跟服务器不太一样了！工作站常常仅针对内部的几个使用者开放而已， 通常是不希望连上 Internet 的啦！而且所谓的工作站自然就是用来做工的！ 例如鸟哥的其中一部 Linux 就是专门用来进行大型的数值模式计算仿真之用！ 这个时候的远程联机服务器可能就得要对多人启动了！ 因为工作站的强大运算功能可以让很多人一同使用他的计算能力！而且也可以免除每部计算机都得要安装 compiler 的窘境！要知道，某些工程用的 compiler 是粉贵的～

有哪些可供登入的类型？

那么目前远程联机服务器的主要类型有哪些？如果以显示的类型来分类， 基本上有文字接口与图形接口两种。

在文字类型登入方面的服务器，主要有以『明码』传送数据的 telnet 服务器，及以加密技术进行封包加密来传送的 SSH 服务器！虽然 telnet 可以支持的 client 端软件比较多，不过由于他是使用明码来传送数据， 您的数据很容易遭到有心人士的撷取！所以近来我们都呼吁大家多使用 SSH 这一种联机方式，而舍弃掉 telnet 这个比较不安全的咚咚啰！

至于图形接口的服务器，比较简单的有 Xdmcp ，架设 Xdmcp 很简单， 不过 client 端的软件比较少。另外一款目前很常见的服务器，就是 VNC (Virtual Network Computing)， 透过 VNC server/client 软件来进行连接。图形接口最大的优点是『图形』啊！ 不过，因为是透过图形来传送，传输的数据量相当的大，所以速度与安全性都有待考虑。 因此，我们仅建议您将图形接口的远程登入服务器开放在内部网域 (LAN) 就好了！

那么什么是『明码』与『加密』的数据封包传送模式呢？ 为什么 telnet 使用明码就比较不安全？所谓的明码就是： 『当我们的数据封包在网络上流窜时，该数据封包的内容为数据的原始格式』， 还记得我们在网络常用指令章节当中介绍的 tcpdump 吧？ 我们在 telnet 下达的指令与密码等等，都会以类似 ASCII 的格式传送到主机端， 而主机端就藉由这些数据来下达指令。如果这些数据封包在经过某些 broadcast 或者是 Router 时，被有心人士捉去，那么他将会完整的取得您的数据喔！ 所以啦，万一您的数据封包里面含有信用卡数据、密码、身份确认等重要信息时，是否很危险呐？！ 因此，目前我们通常都希望使用可以将这些在网络上面跑的数据加密的技术，以增加数据在 Internet 上面传送的安全性啊！

Telnet 服务器

知道 telnet 是什么吗？咦！不就是连接 BBS 的工具吗？没错！他确实也是 BBS 软件类的一个服务器啦！不过这里我们暂不玩弄 BBS ！ telnet 可以说是历史相当悠久的远程联机服务器哩！而且支持他的软件也相当的多！例如知名的 netterm 就直接支持他啦！联机之后的界面也漂亮，在 client 端的中文传输与输入也没有问题！ 相当的不错用！不过，他最麻烦的地方就是.....比较不安全而已啦～

底下我们谈一谈怎么启动与使用 telnet 服务器吧！

安装、启动与关闭服务

- 安装：

近年来由于 telnet 是以明码在传输的问题，所以在新的 Linux 版本上面，已经都将 telnet 这个服务器排除在『先发名单』之外啦，也就是说，很多 Linux distributions 预设是不安装 telnet 的，不过，在每个主要的 Linux distributions 还是有提供 telnet 套件在光盘当中啦！所以您要拿出原版光盘，并且安装好他就可以用啦！如何确认是否已经安装了 telnet 呢？最简单的方法就是使用最广泛被使用的 RPM 啦！

```
[root@linux ~]# rpm -qa | grep telnet
telnet-0.17-31.EL4.3
telnet-server-0.17-31.EL4.3
# 上面是 CentOS 4.x 预设的套件版本。如果是其它的 distribution,
# 档名可能会不太一样～可利用 yum 或 apt 等方式来安装喔!
```

需要特别留意的是，如果要提供 telnet 联机服务，通常需要安装两个 RPM 喔：

1. 一个是 telnet，这个套件提供的是 telnet 客户端的联机程序；
2. 另一个是 telnet-server 套件，这个才是真正的 Telnet server 软件喔！

如果找不到 telnet-server 的话，请拿出原版光盘来安装，或者直接使用 yum 吧！ 否则就无法进行下一步的设定啦！^_^

- 启动与关闭：

还记得『鸟哥的 Linux 私房菜 -- 基础学习篇』里面的『认识服务(daemon)』那个章节吧？ 要记得 super daemon 呦！因为我们的 telnet 就是挂在 super daemon 底下的一支服务而已！那个咚咚就是有名的 xinetd 啰！

> Tips:
> 在某些旧版的套件上面也有使用 inetd 的，启动的方式有点不太一样，不过差异不大啦！ 只要懂得基本的常识，那么就不会有问题啰！所以鸟哥才会要大家先读完 Linux 基础篇 啦！



启动的方式就是：

1. 将 xinetd 里面关于 telnet 的项目开启，然后
2. 重新启动一次 xinetd 就成功啦！

那么如何开启 telnet 的项目呢？很简单，有两个方式：

1. 使用 ntsysv 或 chkconfig： 还记得 Red Hat 系列(含 CentOS)的套件里面的 ntsysv 这个好用的东西吗？对了，在 Fedora 底下有这么一个好用的设定工具，您可以使用 ntsysv 出现的窗口之中，将 telnet 勾选起来，然后按下 OK 离开即可啰！

2. 使用 vi 修改 /etc/xinetd.d/telnet 这个档案： 那么如果不是 Red Hat 的 Linux 系统呢？基本上， ntsysv 也只是修改 /etc/xinetd.d 这个目录下的数据而已，所以我们当然可以手动自己修改他啦！

```
[root@linux ~]# vi /etc/xinetd.d/telnet
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
#       disable         = yes
        disable         = no
# 基本上，改上面这两行就够了！将 disable 设定成 no 表示要启动！
}
```

设定完开启之后，自然就是要启动啦，刚刚提到 telnet 是挂在 xinetd 底下的，所以自然只要重新启动 xinetd 就能够将 /etc/xinetd.d/ 里头的设定重新读进来， 所以刚刚设定启动的 telnet 自然也就可以被启动啦！而启动的方式也有两种方式，其中 service 这个指令仅支持在 CentOS 与 Mandriva 底下，所以通常鸟哥还是以 /etc/init.d 底下的 scripts 为启动的主要方法啦！

```
仅适合 Red Hat 系列 / Mandriva 系列的主机启动方式
[root@linux ~]# service xinetd restart
Stopping xinetd:                                [  OK  ]
Starting xinetd:                                [  OK  ]

适合各版本的主机启动方式
[root@linux ~]# /etc/init.d/xinetd restart
Stopping xinetd:                                [  OK  ]
Starting xinetd:                                [  OK  ]
# 某些版本并没有 restart 的选项，这个时候就需要：stop 再 start 啰！
```

那么要看有没有启动服务呢？怎么看？其实也很简单啦，还记得我们在前几章提到的『 限制 Linux port 的联机 』那一章吗？使用 netstat 就可以啦！

```
[root@linux ~]# netstat -tlup
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address    State   PID/Program name
tcp        0      0 *:telnet       *:*                LISTEN  23817/xinetd
```

看到了吗？没错，那个 telnet 就是启动的项目啦！那么要如何关闭呢？呵呵！ 那就真的是太简单啦！就将刚刚的步骤再做一次，而将设定值转变一下即可！步骤如下啦！

Tips:
这里考一个问题，那个 port 对应的服务名称在哪一个档案里面查询到的呢？ 在每一个 Linux 系统都有的档案呦！忘记了呀！？再回到前面看看 限制 Linux port 的联机 ， 然后用 vi 去看看那一个档案的内容吧！ ^_^



```
Step 1: 修改设定档
[root@linux ~]# vi /etc/xinetd.d/telnet
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        disable         = yes  <== 就是这里啦！将他改成 yes 就是关闭！
}

Step 2: 重新启动 xinetd 这个 super daemon
[root@linux ~]# /etc/init.d/xinetd restart
```

好用的联机软件

刚刚上面提到的都是在服务器端的设定而已！那么在客户端有什么好用的软件可以连上 Server 的呢？最常见到的应该就是 netterm 这个鼎鼎大名的联机软件了吧！ 我想，只要玩过 BBS 的大概都晓得这个软件才对！所以这里就不提了！ 另外，目前几乎所有的操作系统都提供了 telnet 这个程序，这个程序可以直接轻易的就连上 telnet server 呢！例如您要在 Linux 上面连上自己的 telnet 服务器，可以这样做：

```
[root@linux ~]# telnet localhost
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
CentOS release 4.4 (Final)
Kernel 2.6.9-42.0.2.EL on an i686
login: dmtsai    <== 就是这里啦！请输入『一般』账号，不能用 root 喔！
Password:         <== 这里输入该账号的密码！请注意，输入时，屏幕不会有任何信息！
Last login: Fri Jul  1 09:31:21 from 127.0.0.1  <== 上次登入的信息为何?
[dmtsai@linux ~]$ <== 这里就是已经登入的地方！亦即远程主机了！
[dmtsai@linux ~]$ exit    <== 这样就能够离开 telnet 与远程主机咯！
```

这样就联机进来啦！很简单吧！那么在 Windows 的环境底下呢？同样的，也是可以使用 telnet 的程序联机到 Linux 的 telnet server 里面来！没有问题的啦！可以依序这样做：

1. 按下 Windows 内的 『开始』
2. 选择 『执行』
3. 在出现的窗口中输入 『telnet your.IP.or.hostname』

这样就可以进入 Linux 的环境中了！很方便吧！当然啦！您也可以使用类似 netterm 这个很棒的联机软件来联机的，这里我们就不示范啦！

> Tips:
> 在 Linux tty1 ~ tty6 的终端机预设模式下，我们是没有办法看到中文的！ 除非安装某些特殊的中文接口才行！比如 JMCCE 之类的咚咚！ 因为不是很重要，所以鸟哥这里就不加介绍了。 ^_^



另外，需要先留意的是，为了系统安全的考虑，预设的 telnet 是『不允许』使用 root 这个账号登入的～ 这个很重要喔！您不要使用 root 尝试登入 telnet 啊！ ^_^

---

iptables, TCP_Wrappers, 纯建议

telnet 这个服务器方便归方便，但总是一个不太好的联机解决方案， 因为毕竟他是一个以『明码』传输的协议，所以很不适合在 Internet 上面使用啦！ 你总不希望你的账号密码在 Internet 上面被窃取吧？ 不过，如果 telnet 是启动在内部环境当中那就还好啦！ 尤其有些朋友因为旧软件的关系，还是需要使用到 telnet 来联机。那么我们就提一些基本的注意事项好了！

- 以比较限制的设定档来规范联机的 IP ：

事实上， xinetd 就已经提供些许的保护措施了， 您可以针对您主机的多重接口(有对内以及对外喔！)来提供不同保护等级的措施！ 底下列出一个范例，不过，更多的信息请再回到『鸟哥的 Linux 私房菜 -- 基础学习篇』当中去查阅一下 『认识服务 』那一章里面的详细设定说明，或者直接 man xinetd.conf 吧！

```
[root@linux ~]# vi /etc/xinetd.d/telnet
# This file had been modified by VBird 2002/11/04
# First is about inside the network
service telnet
{
        disable         = no
        bind            = 192.168.1.2
        only_from       = 192.168.1.0/24
        # 上面这两行说明仅提供内部网域！
        instance        = UNLIMITED
        nice            = 0
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
```

```
        server          = /usr/sbin/telnetd
        server_args     = -a none
        log_on_failure  += USERID
}

# Second is about the outside domain's settings
service telnet
{
        disable         = no
        bind            = 140.116.142.196
        only_from       = 140.116.0.0/16
        no_access       = 140.116.32.{10,26}
        # 上面这三行设定外部较为严格的限制
        instance        = 10   <==最多允许同时 10 个联机
        umask           = 022
        nice            = 10
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/telnetd
        server_args     = -a none
        log_on_failure  += USERID
}
```

- root 不能直接以 telnet 连接上主机：

既然 telnet 不是很安全，自然预设的情况之下就是无法允许 root 以 telnet 登入 Linux 主机的！但事实上， telnet 只是利用一些较为安全的机制（其实就是 PAM 模块啦）来防止 root 登入而已～所以啰，假如您确定您的环境够安全(例如您的主机并没有连上 Internet)，并且想要开放 root 以 telnet 登入 Linux 主机的话，请直接将 /etc/securetty 更改檔名即可！

```
[root@linux ~]# mv /etc/securetty /etc/securetty.bak
```

这样一来，root 就可以登入啦！不过，相当的不建议这样做喔！毕竟不是很安全啦！此外，您也可以藉由修改 pam 模块来达成同样的功能！修改 /etc/pam.d/login 这个档案的第二行设定即可：

```
[root@linux ~]# vi /etc/pam.d/login
#%PAM-1.0
#auth       required     pam_securetty.so  <== 就是这样一行，将他批注即可！
auth       required     pam_stack.so service=system-auth
auth       required     pam_nologin.so
account    required     pam_stack.so service=system-auth
password   required     pam_stack.so service=system-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_stack.so service=system-auth
```

```
session     required      pam_loginuid.so
session     optional      pam_console.so
# pam_selinux.so open should be the last session rule
session     required      pam_selinux.so multiple open
```

如此一来， root 将可以直接进入 Linux 主机了！不过，既然我们可以透过 su 或 sudo 来切换身份， 那么干嘛还需要开放 root 用 telnet 登入主机呢？真是没必要～所以， 还是不建议如此做的！

- 加上防火墙 iptables：

针对 telnet 加设防火墙 iptables 是一个好主意！如果您已经参考了前面章节提到的『 简易防火墙架设 』一文，并且使用里面的 scripts 的话，那么不用担心 telnet 啦！基本上，他原本就仅对内部开放 telnet ，外部是无法连上您的 telnet 的！但是，若是您自己设定了自己的防火墙机制之后，那么想要针对 192.168.0.0/24 这个网域，及 61.xxx.xxx.xxx 这个 IP 进行 telnet 的开放呢？可以增加这几行在您的 iptables 规则之内(请注意：防火墙的规则顺序是很重要的！所以再回头看看 简易防火墙架设 一文是有必要的！)

```
iptables -A INPUT -p tcp -i $INIF  -s 192.168.0.0/24 --dport 23 -j ACCEPT
iptables -A INPUT -p tcp -i $EXTIF -s 61.xxx.xxx.xxx --dport 23 -j ACCEPT
iptables -A INPUT -p tcp -i $EXTIF                   --dport 23 -j DROP
```

上面的规则中，$EXTIF 指的是对外的联机接口，$INIF 则是对内的接口。 第一、二行是针对来源的 IP 来开放 port 23 亦即是 telnet 的协议啦！ 而最后一行则是将其它的所有来源的，想要连上 telnet 的联机封包都丢掉的意思！怎么样！很简单吧！

- 加上防火墙 /etc/hosts.allow(deny) 机制：

防火墙的机制是越多越好！永远也不嫌多的啦！这里也可以使用 TCP_Wrappers 的机制呢！刚刚是开放了 192.168.0.0/24 这个网段，但是如果您只想要其中的 192.168.0.1 ~ 192.168.0.5 进入呢？而其它的 IP 只要一经联机，就会被记录该 IP ，以提供 root 查询呢？可以这样做：

```
[root@linux ~]# vi /etc/hosts.allow
in.telnetd: 192.168.0.1, 192.168.0.2, 192.168.0.3, 192.168.0.4
in.telnetd: 192.168.0.5

[root@linux ~]# vi /etc/hosts.deny
in.telnetd : ALL : spawn (/bin/echo Security notice from `/bin/hostname`; \
/bin/echo; /usr/sbin/safe_finger @%h ) | \
/bin/mail -s "%d -%h security" root@localhost & \
: twist ( /bin/echo -e "\n\nWARNING connectin not allowed. \n\n\n" )
```

- 建议事项：

事实上， telnet 最大的不安全在于数据是以明码传输，所以在 Internet 这个大家都能够连上的地方来传输数据时，实在很不安全！所以：

1. 非必要时，不要启动 telnet ，如果真的需要启动 telnet ，那么也请在启动并且使用完毕之后，立即将他关掉！
2. 如果确定真的要启动 telnet 时，请确定好限制的联机范围，使用 iptables 来设定联机的限制区域；

3. 加上 TCP_Wrappers 的辅助，加强防火墙的功能！
4. 随时注意登录档案里面关于 login 的事项！并且不要让 root 能以 telnet 登入 Linux 主机！

SSH 服务器

既然 telnet 的数据在 Internet 上不是很安全，那么我又需要以远程联机服务来操控我的 Linux 主机，那么应该怎么办呀？最好的方法当然就是以较为安全的联机机制来解决联机的问题啰！那么该如何解决这样的问题呢？这也不难啦，使用 SSH 即可。那么 SSH 是什么呢？他有什么特异功能？

简单的来说，SSH 是 Secure SHell protocol 的简写，他可以经由将联机的封包加密的技术， 来进行数据的传递，因此，数据当然就比较安全啰！这个 SSH 可以用来取代 Internet 上面较不安全的 finger, R Shell (rcp, rlogin, rsh 等指令), talk 及 telnet 等联机模式。底下我们将先简介一下 SSH 的联机模式，来说明为什么 SSH 会比较安全呢！

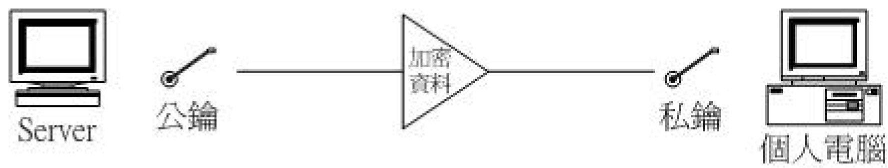
特别注意：这个 SSH 协议，在预设的状态中，本身就提供两个服务器功能：

1. 一个就是类似 telnet 的远程联机使用 shell 的服务器，亦即是俗称的 ssh ；
2. 另一个就是类似 FTP 服务的 sftp-server ！提供更安全的 FTP 服务。

联机加密技术简介：

什么是『数据加密』呢？简单的说，就是将人们看的懂得电子数据，经过一些运算， 让这些数据变成没有意义的(至少对人类来说)咚咚，然后这个咚咚可以在网络上面传输， 而当使用者想要查阅这个数据时，再透过反向运算，将这些咚咚反推出原始的电子数据。 由于这些数据已经被重新处理过，所以，即使数据在 Internet 上被 cracker 监听而窃取，他们也不容易就推算得出来原始资料内容的。

网络封包的加密技术通常是藉由所谓的『一对公钥与私钥』 亦即『Public and Private 组合成的 key pair』来进行加密与解密的动作！如下图所示。主机端所要传给 client 端的数据，会先经由公钥加密后才到网络上传输。 而到达 client 端之后，再经由私钥将加密的数据解开来～由于在 Internet 上面跑的数据是加密过后的， 所以你的数据内容当然就比较安全啦！



图一、公钥与私钥在进行数据传输时的角色示意图

Tips:
数据加密的技术真的相当的多,也各有其优缺点,有的运算速度快,但是不够安全;有的够安全,但是加密/解密的速度较慢～ 目前在 SSH 使用上,主要是利用 RSA/DSA/Diffie-Hellman 等机制喔!



那么这些公钥与私钥是如何产生的呢?底下我们来谈一谈目前 SSH 的两种版本的联机模式啰!

- SSH protocol version 1:

每一部 SSH 服务器主机都可以使用 RSA 加密方式来产生一个 1024-bit 的 RSA Key , 这个 RSA 的加密方式,主要就是用来产生公钥与私钥的演算方法!这个 version 1 的整个联机的加密步骤可以简单的这么看:

1. 当每次 SSH daemon (sshd) 启动时,就会产生一支 768-bit 的公钥(或称为 server key)存放在 Server 中;
2. 若有 client 端的 ssh 联机需求传送来时,那么 Server 就会将这一支公钥传给 client ,此时 client 也会比对一下这支公钥的正确性。比对的方法为利用 /etc/ssh/ssh_known_hosts 或 ~/.ssh/known_hosts 档案内容。
3. 在 Client 接受这个 768-bit 的 server key 之后,Client 自己也会随机产生一支 256-bit 的私钥(host key),并且以加密的方式将 server key 与 host key 整合成一对完整的 Key pair,并且将这对 Key pair 也传送给 server ;
4. 之后,Server 与 Client 在这次的联机当中,就以这一对 1024-bit 的 Key pair 来进行数据的传递!

也就是说,Public Key 是放在 Server 上的,而 Client 端的软件必须要能接受 Public Key 以及计算出 Private Key 以组合成为一把独一无二的 key pair ,因为 Client 端每次的 256-bit 的 Key 是随机取的,所以您这次的联机与下次的联机的 Key 可能就会不一样啦!此外在 Client 端的使用者家目录下的 ~/.ssh/known_hosts 会记录曾经联机过的主机的 public key ,用以确认每次来自该主机的联机是正确的。 这个 ~/.ssh/known_hosts 档案的意义后续还会介绍的。

- SSH protocol version 2:

在 SSH version1 的联机过程当中,当 server 端接受 client 端的 private key 后,就不再针对该次联机的 key pair 进行检验。此时若有恶意的 cracker 针对该联机给予恶意的程序代码时,由于主机端不会检验联机的正确性, 因此可能会接受该程序代码,进一步造成系统被黑掉的问题。

为了改正这个缺失,SSH version 2 多加了一个确认联机正确性的 Diffie-Hellman 机制, 在每次数据的传输当中 server 端都会以该机制检查资料的来源是否正确, 所以可以避免联机过程当中被插入恶意程序代码的问题! 也就是说, ssh version 2 是比较安全的喔!

由于 SSH version 1 本身存在着的一些问题,因此,近来我们都希望大家使用 ssh version 2 的联机模式, 会比较安全一点。而联机版本的设定则需要在 ssh 主机端与客户端均设定好才行喔!

启动 SSH 服务:

事实上，在我们使用的 Linux 系统当中，预设就已经含有 SSH 的所有需要的套件了！ 这包含了可以产生密码等协议的 OpenSSL 套件与 OpenSSH 套件，所以呢，要启动 SSH 真的是太简单了！就直接给他启动就是了！此外，在目前的 Linux Distributions 当中，都是预设启动 SSH 的， 所以一点都不麻烦，因为不用去设定，他就已经启动了！哇！真是爽快～无论如何， 我们还是得说一说这个启动的方式吧！直接启动就是以 SSH daemon ，简称为 sshd 来启动的，所以，手动可以这样启动：

```
[root@linux ~]# /etc/init.d/sshd restart
[root@linux ~]# netstat -tlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address   State    PID/Program name
tcp        0      0 *:ssh           *:*               LISTEN   24266/sshd
```

启动后，利用 netstat 查阅一下 sshd 这个程序是否正确的在 LISTEN 即可！当然，这个时候您的 SSH 服务器设定值均是使用系统默认值， 能不能够仅用较安全的 version 2 ，则需要进一步的设定呢。 接下来，如果您想要在开机就启动 SSH 的话(预设也是启动的！)，可以利用 chkconfig 来设定开机启动即可。

虽然新的 Linux distributions 都预设会有 SSH 存在的，但是较旧的版本就仅有 telnet 而已。 例如 Red Hat 6.x 之前的版本。那么如果您想要在旧的 distributions 当中安装 SSH 该如何是好？ 嘿嘿！可以参考一下鸟哥之前写过的一篇咚咚，有详细的说明 tarball 的安装流程哩！

使用 Tarbal 安装 SSH 以及升级 SSH 可能会遇到的问题说明
(http://linux.vbird.org/linux_server/0310telnetssh/0310telnetssh-2.php)

需要注意的是， SSH 不但提供了 shell 给我们使用，亦即是 ssh protocol 的主要目的，同时亦提供了一个较为安全的 FTP server ，亦即是 ssh-ftp server 给我们当成是 FTP 来使用！所以，这个 sshd 可以同时提供 shell 与 ftp 喔！而且都是架构在 port 22 上面的呢！所以，底下我们就来提一提，那么怎么样由 Client 端连接上 Server 端呢？同时，如何以 FTP 的服务来连接上 Server 并且使用 FTP 的功能呢？

---

ssh 客户端联机：

由于 Linux 与 Windows 这两个客户端 Client 联机软件/指令并不一样，所以我们分别来介绍可以使用的指令：

- Linux Client: ssh

SSH 在 client 端使用的是 ssh 这个指令，这个指令可以指定联机的版本 (version1, version2)， 还可以指定非正规的 ssh port (正规 ssh port 为 22)。不过，一般的用法可以使用底下的方式：

```
1. 直接登入到对方主机的方法：
[root@linux ~]# ssh account@hostname
# 连接到我们自己本机上面的 ssh 服务！更多讯息，请 man ssh 喔！
[root@linux ~]# ssh dmtsai@localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is f8:ae:67:0e:f0:e0:3e:bb:d9:88:1e:c9:2e:62:22:72.
Are you sure you want to continue connecting (yes/no)? yes
# 上面很重要喔！务必填入完整的 "yes" 而不是 Y 或 y 而已。
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
```

```
dmtsai@localhost's password: <== 在这里填入密码，同样的，屏幕不会有讯息的!
Last login: Fri Jul  1 14:23:27 2005 from localhost.localdomain
[dmtsai@linux ~]$      <== 瞧! 已经登入啰~
[dmtsai@linux ~]$ exit <== 输入 exit 就能够离开对方主机啰!

2. 不登入对方主机，直接在对方主机执行指令的方法:
[root@linux ~]# ssh dmtsai@localhost date
dmtsai@localhost's password:
Tue Nov 22 11:57:27 CST 2005
[root@linux ~]#
# 看! 身份还是 root 喔! 只是以 dmtsai 的身份在远程主机上执行了一个指令而已!
```

这里请特别留意的是，如果直接以『 ssh hostname 』这个指令来连接进入 hostname 这个主机时，则进入 hostname 这个主机的『账号名称』将会是目前您所在的这个环境当中的使用者账号！ 以上面为例，因为我是以 root 的身份在执行，所以如果我执行了『 ssh host.domain.name 』时， 那么对方 host.domain.name 这部主机，就会以 root 的身份来让我进行密码确认的登入动作！

因此，为了避免这样的麻烦，通常鸟哥都是以简单的 e-mail 的写法来登入远方的主机， 例如『ssh user@hostname 』即表示， 鸟哥是以 user 这个账号去登入 hostname 这部主机的意思。当然，也可以使用 -l username 这样的形式来书写！登入对方主机之后，其它的所有执行行为都跟在 Linux 主机内没有两样~所以，真的是很简单吧！ ^_^ 这样就可以达到远程控管主机的目的了！

此外，在预设的情况下， SSH 是『允许您以 root 的身份登入』喔 ！呵呵！更是爽快啦！要特别留意的是，当您要连接到对方的主机时，如果是首次连接，那么 Server 会问您，您的联机的 Key 尚未被建立，要不要接受 Server 传来的 Key ,并建立起联机呢？呵呵!这个时候请『务必要输入 yes 而不是 y 或 Y』，这样程序才会接受喔！

- 关于 Server Keys 的纪录数据： ~/.ssh/known_hosts

如果您刚刚有研究过 SSH 的联机流程的话，会发现到当 client 端接受来自 server 端的 public key 之后，会主动的比对这支 Key 的正确性。而比对的档案是 ~/.ssh/known_hosts 。若是接受到的这支 public key 并没有被纪录在这档案内，那么上面表格的讯息， 就是要您回答 yes/no 的那个讯息才会出现~而您回答 yes 之后，该 public key 信息就会被记录起来，以留待下次登入同一部主机时的检查之用啊！ 如果 Server Key 与 ~/.ssh/known_hosts 比对成功， 那么您就会直接进入等待密码输入的画面， 那就不必每次都得要输入 (yes/no) 啰~

不过，您或许也会发现一件事情啊，我们知道 SSH server 虽然使用 version 2 已经不会重复制造 server key (public key) 了，但是如果该主机重新安装过新的 linux distributions 时， 那把 server key 就会被改变啊！而 client 又会去比对这个 public key 与 ~/.ssh/known_hosts ， 此时 Client 就会发现两者不同了，于是乎产生如下的错误讯息了：

```
[root@linux ~]# ssh dmtsai@localhost
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
```

```
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
f8:ae:67:0e:f0:a0:3e:aa:d9:77:19:c9:2e:62:22:72.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending key in /root/.ssh/known_hosts:1
RSA host key for localhost has changed and you have requested strict checking.
Host key verification failed.
```

这个错误讯息在告诉您，上次所登录的远程 SSH 主机的 Keys 已经被改过了(最可能的原因就是 Server 端重新开机/重新安装/更新套件等等啦！)， 所以无法继续登入～呵呵！这个时候怎么办？很简单啊！进入您的家目录的 ~/.ssh 里面，编辑一下 known_hosts ，将欲连接的主机名称的 Key 给他消除，就可以重新联机啦！

```
[root@linux ~]# vi ~/.ssh/known_hosts
localhost ssh-rsa AAAAB3NzaC1yc2Euowireffodjoiwjefmoeiwhoqhwupoi
t[egmlomowimvoiweo6VpTHTw2/tENp4U7Wn8J6nxYWP36YziFgxtWu4MPSKaRmr
E4eUpR1G/zV3TkChRZY5hGUybAreupTVdxCZvJlYvNiejfijoejwiojfijeoiwx5
eRkzvSj7a19vELZ5f8XhzH62E=
```

上面表格的内容其实是同一行的～那一行代表『 localhost 这部主机，利用的是 ssh-rsa 联机机制， 而后续的数据则是那把 Server Key 的内容。』如果您确定这次的比对无法成功是正常的， 那么您可以将这一行给他删除，这样下次要再登入时，就又会再次的出现问 (yes/no) 的画面啊！

- Linux Client: sftp

那么如何使用 SSH FTP 的功能呢？也是很容易啦！就是使用 sftp 这支程序即可！而登入的方式与 ssh 相同，都是使用 sftp -l username hostname 或者直接以 sftp user@hosname 来书写！执行之后会有底下的模样：

```
[root@linux ~]# sftp dmtsai@localhost
Connecting to localhost...
dmtsai@localhost's password: <== 这里请输入密码啊！
sftp> <== 这里就是在等待您输入 ftp 相关指令的地方了！
```

进入到 sftp 之后，那就跟在一般 FTP 模式下的操作方法没有两样了！底下我们就来谈一谈， sftp 这个接口下的使用指令吧！

针对远方主机(Server)之行为	
变换目录到 /etc/test 或其它目录	cd /etc/test cd PATH
列出目前所在目录下的文件名	ls dir
建立目录	mkdir directory

删除目录	rmdir directory
显示目前所在的目录	pwd
更改档案或目录群组	chgrp groupname PATH
更改档案或目录拥有者	chown username PATH
更改档案或目录的权限	chmod 644 PATH 其中，644 与权限有关！回去看基础篇！
建立连结档	ln oldname newname
删除档案或目录	rm PATH
更改档案或目录名称	rename oldname newname
离开远程主机	exit (or) bye (or) quit
针对本机(Client)之行为(都加上 l, L 的小写)	
变换目录到本机的 PATH 当中	lcd PATH
列出目前本机所在目录下的文件名	lls
在本机建立目录	lmkdir
显示目前所在的本机目录	lpwd
针对资料上传/下载的行为	
将档案由本机上传到远程主机	put [本机目录或档案] [远程] put [本机目录或档案] 如果是这种格式，则档案会放置到目前远程主机的目录下！
将档案由远程主机下载回来	get [远程主机目录或档案] [本机] get [远程主机目录或档案] 若是这种格式，则档案会放置在目前本机所在的目录当中！可以使用万用字符，例如： get * get *.rpm 亦是可以的格式！

就整体而言， sftp 在 Linux 底下，如果不考虑图形接口，那么他已经可以取代 FTP 了呢！因为所有的功能都已经涵盖啦！因此，在不考虑到图形接口的 FTP 软件时，可以直接关掉 FTP 的服务，而改以 sftp-server 来提供 FTP 的服务吧！ ^_^

- Linux Client: scp

如果我要在两个主机之间复制档案的话，除了 sftp 之外，还有没有更简单的方式？ 有的，那就是利用 scp 这个指令啦！这个指令的用法与 cp 很相像，不过， 在远程主机的目录写法，比较需要注意就是了。举例如下：

```
1. 将数据由本机上传到远程主机上去
[root@linux ~]# scp /etc/crontab dmtsai@localhost:/home/dmtsai/
dmtsai@localhost's password: <== 这里请输入密码啊！
crontab                  100%   620      0.6KB/s    00:00
# 这个例子在说明，我将本机目录的 /etc/crontab 这个档案传送给 dmtsai
# 这个使用者，而这个使用者是在 "localhost" 那部主机上面喔！
# 仔细看一下，会有一个传输数据的讯息跑出来喔！

2. 将数据由远程主机下载到本机上！
[root@linux ~]# scp dmtsai@localhost:~/.bashrc .
# 这个例子则是在说明，我要将 localhost 那部机器上的 dmtsai 这个人，
# 他家目录下的 .bashrc 复制到我的机器上！
```

也就是说，远程主机上的档案或目录要复制时，是以『 hostname:PATH 』 方式来书写的～不要写错了呦！而如果想要复制目录的话，那么可以加上 -r 的参数！

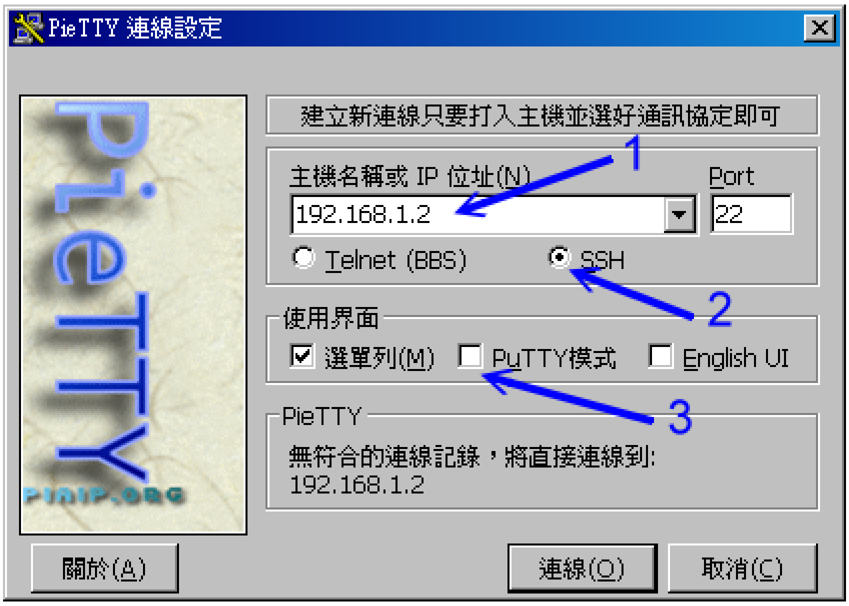
- Windows Client: putty

在 Linux 底下想要连接 SSH 服务器，可以直接利用 ssh 这个指令 ，那么如果在 Windows 操作系统底下，又该如何连接到 SSH 服务器呢？可以直接使用 putty 或 pietty 这种类型的联机软件呢，他也是免费的自由软件喔！取得的方式可以参考底下的网站：

- putty 官方网站：http://www.chiark.greenend.org.uk/~sgtatham/putty/
- pietty 官方网站：http://www.csie.ntu.edu.tw/~piaip/pietty/

在 putty 的官方网站上有很多的 client 软件可以使用的，包括 putty/pscp/psftp 等等。 他们分别对应了 ssh/scp/sftp 这三个指令就是了。而上述的三个 putty/pscp/psftp 主要是在 Windows 上面连接到 Unix like 机器的 SSH 服务器的 Client 软件呢。请自行下载该软件喔。

事实上，鸟哥比较喜欢林弘德先生的 pietty，因为这个软件不但是完整支持 putty ，而且提供的文字编码较丰富， 实在很好用。在你下载了 pietty 后直接双击他， 会有类似底下的图示出现。



图二、pietty 的执行图示之一

在上图 1 的地方请填写相关的主机名称或者是 IP ，2 当然务必选择 SSH 那一项， 至于 3 的地方，鸟哥比较喜欢选单出现的样式，所以我是选择选单啦！若没有问题，按下『联机』后， 就会出现如下样式：

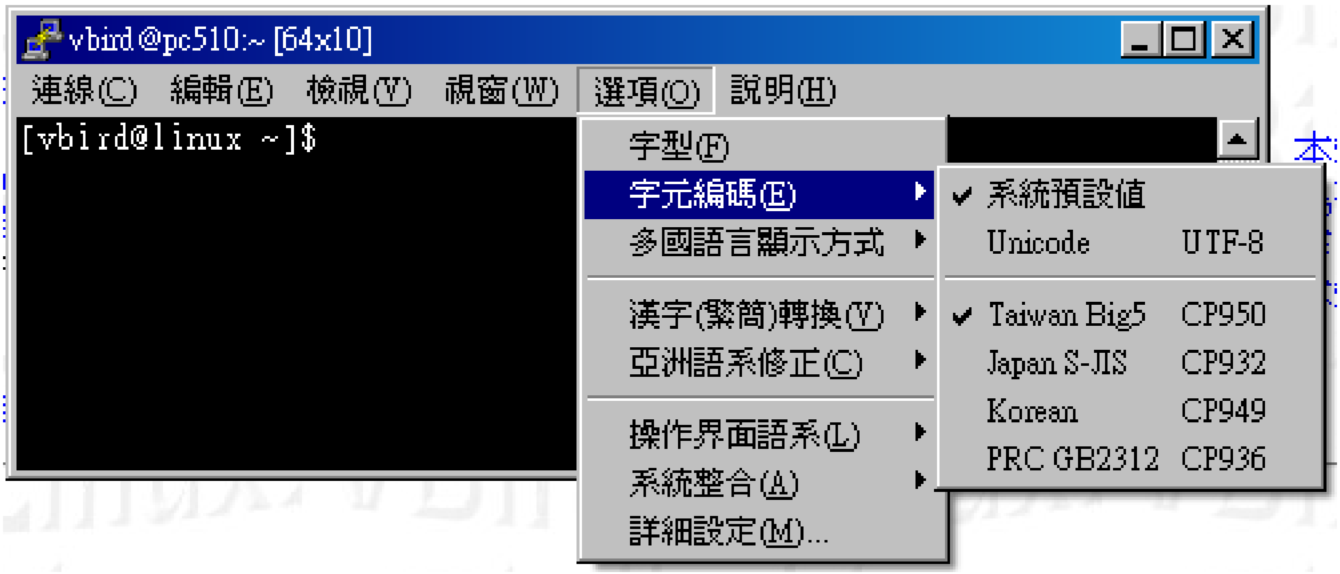


图三、pietty 的执行图示之一

很像在主机前面工作吧！而且上头还有选单可以随时调整类似字形、字体、字符编码等等。 尤其是字符编码。有时候你会发现开启档案时，竟然画面当中会有乱码而不是正常的中文显示， 那就是编码的问题。要解决这个问题时，你必须要牢记：

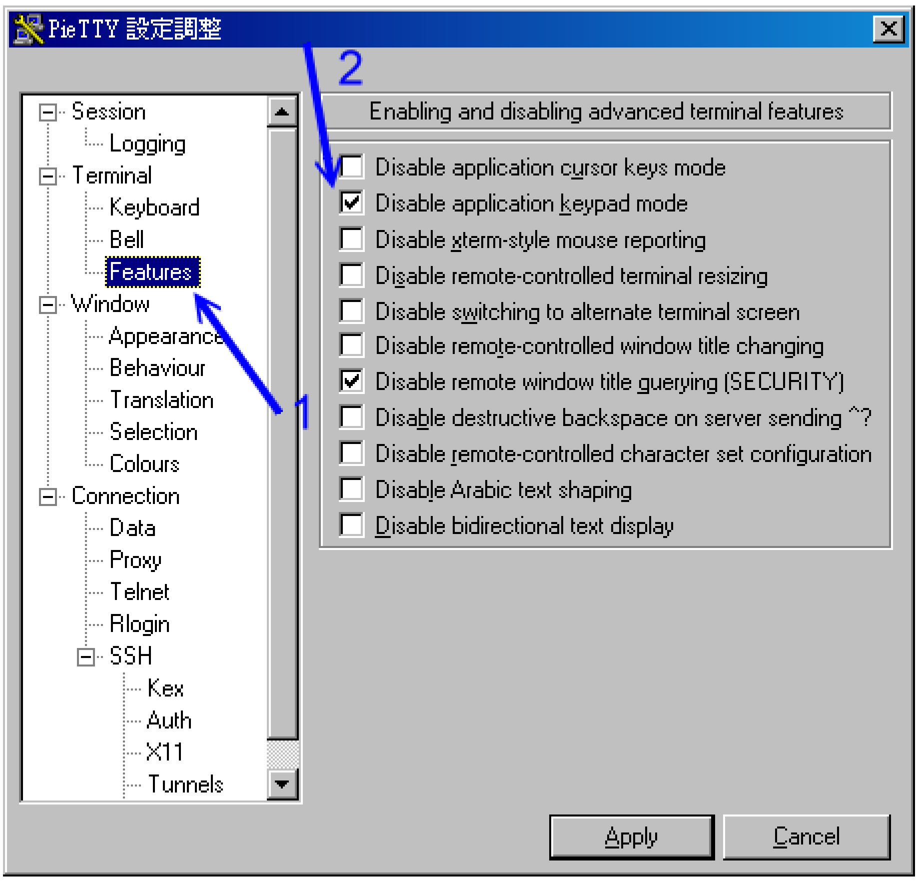
- 文本文件本身在存档时所挑选的语系；
- Linux 主机本身所使用的语系 (可用 LANG 变量调整)；
- pietty 所使用的语系。

这三个咚咚的语系要完全相同时才会正确的显示出中文！千要要牢记啊！ 那如何调整 pietty 的中文编码呢？



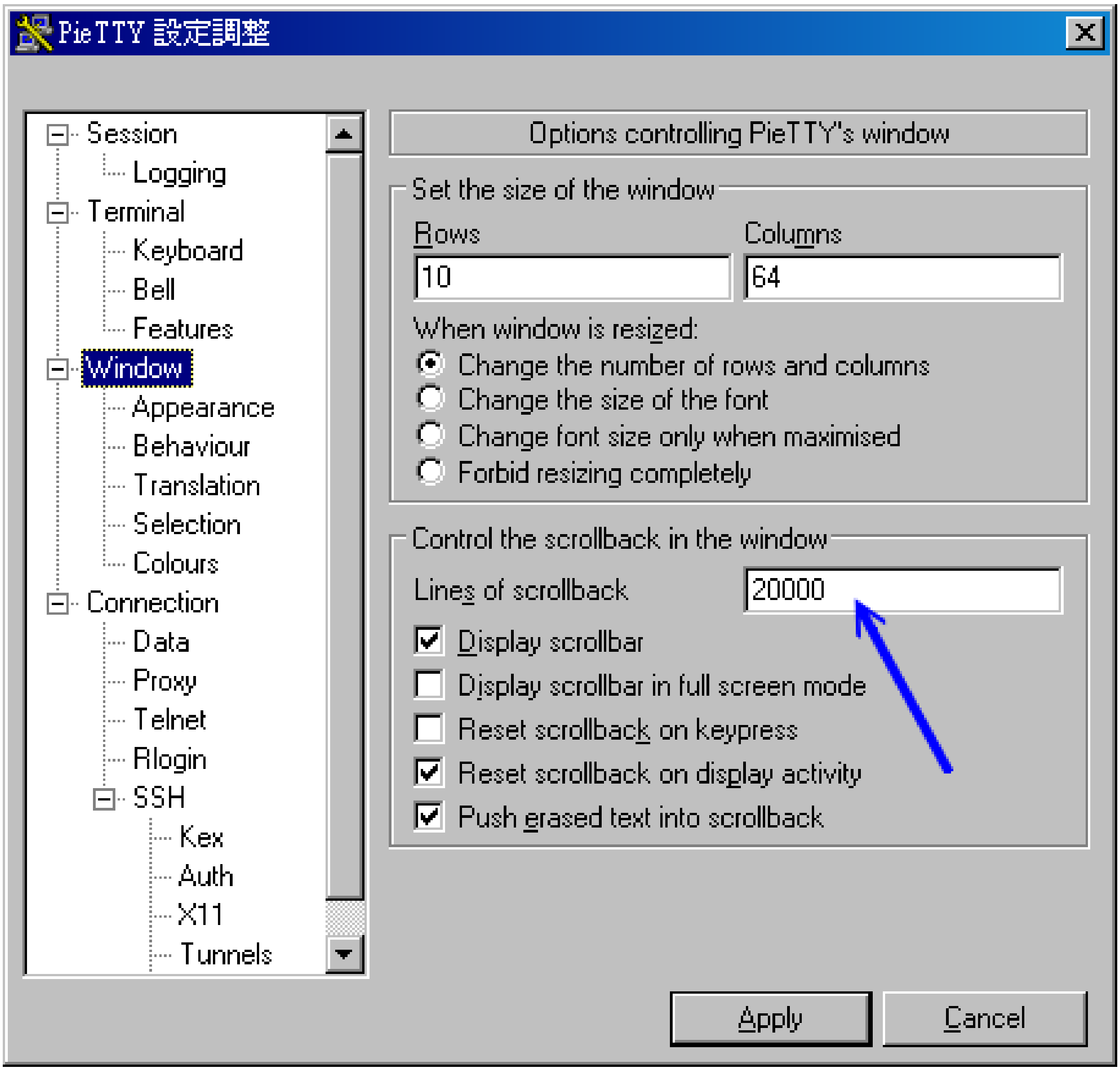
图四、pietty 的执行图示之一

在『选项』的『字符编码』里面可以挑选 big5 或者是 utf8 的中文编码， 让他符合你的 Linux 与档案所储存的数据格式，那就 OK 的啦！ ^_^！ 如果想要作更细部的设定时，可以选择图四上头最底下的那个『详细设定』项目， 就会出现如下图示。其中更为重要的是『键盘右侧的数字键想要生效』时， 可以按照下图的指示来启动数字键的功能：



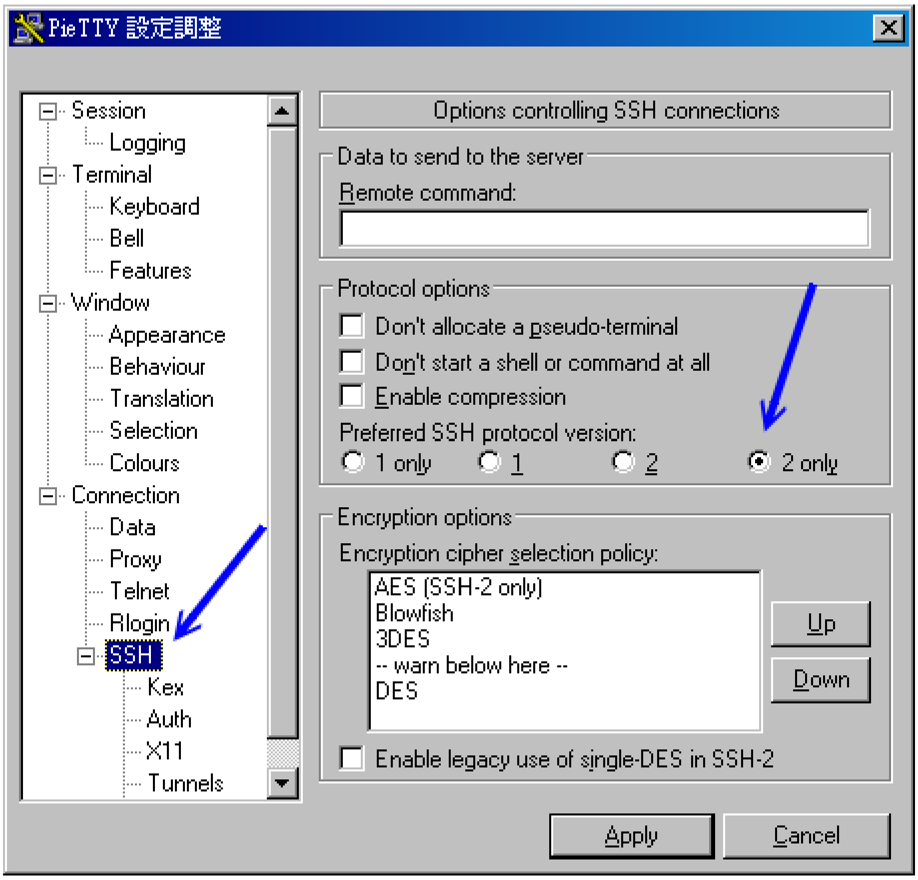
图五、pietty 的执行图示之一

如上图所示，在你输入『Apply』之后，就可以自动的生效了！现在你可以按键盘右边的按钮了， 真方便。再来你可以调整 pietty 滚动条的记忆行数，这样当数据太多时，你依旧可以调整滚动条来查阅之前的数据。 设定的方法如下：



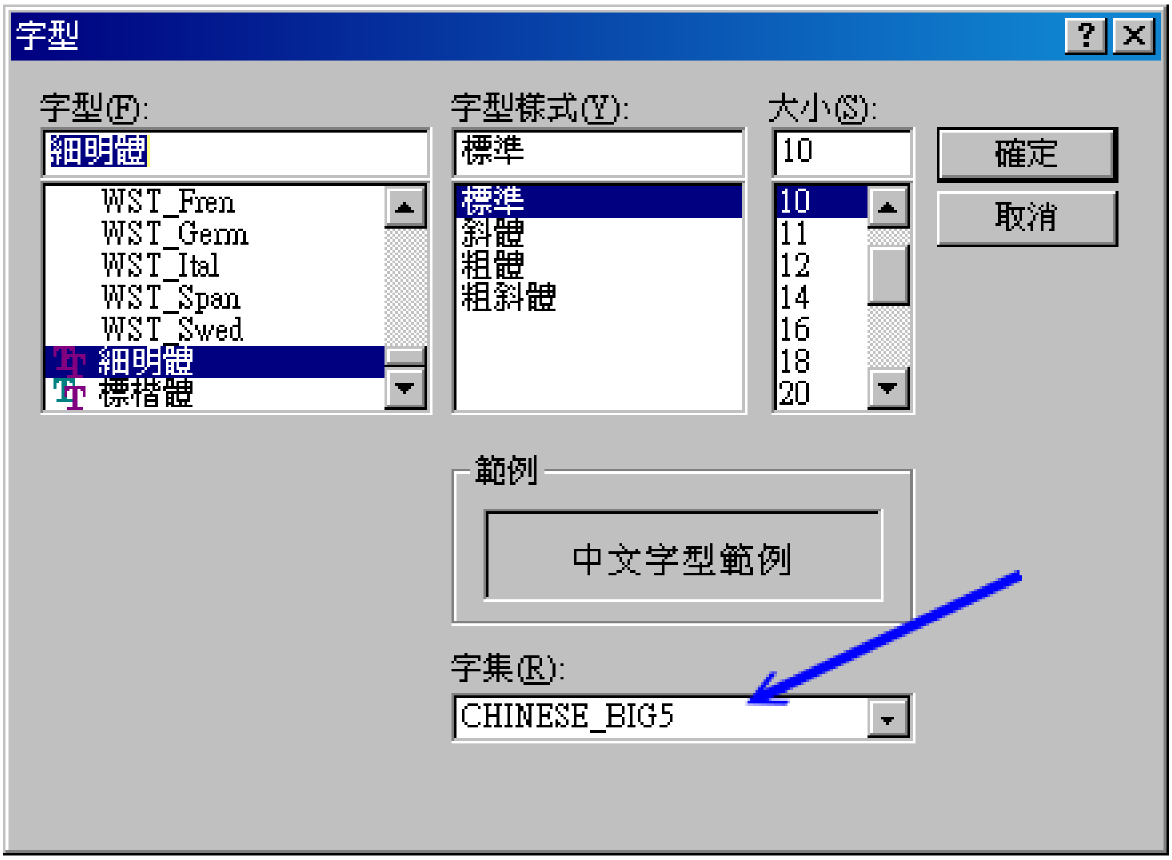
图六、pietty 的执行图示之一

调整完了屏幕的大小之后，再来这是最重要的：『您要以哪一个版本的 SSH 算法登入？！』前面说过，我们预设是以 version2 来登入的，所以这里我们可以调整为 2 那个项目！这样每次登入都会以 version 2 的模式登入主机了！



图七、pietty 的执行图示之一

整个 pietty 大致上的流程就是这样！如此一来，您就可以在 Windows 上面以 SSH 的协议，登入远程的 Linux 主机噜！粉方便吧！ ^_^ ！ 那么如果想要中文支持的话，目前 pietty 已经支持中文啦！您可以输入中文喔！不过需要修改一下字符集， 选择图四的『选项』内的『字型』，会出现如下图示：



图八、pietty 的执行图示之一

将(1)字型设定为细明体 (2)字集设定为『Big5』，如此一来，您的 pietty 就支持中文的输入啰！ ^_^ 那么上面我们作的这些设定值都记录在哪里啊？呵呵！都记录在 Windows 的登录文件当中啊！ 您可以在 Windows 的系统当中，在『开始』--> 『执行』后，出现的框框内输入『regedit』， 之后会出现一个大窗口。请在左边的画面当中选择『 HKEY_CURRENT_USER --> Software --> SimonTatham --> PuTTY --> Sessions』， 就可以看到您的设定值啰！ ^_^！ 这样，也就可以储存您的设定值啰～

- Windows Client: psftp

在 putty 的官方网站上也提供 psftp 这支程序。这一支程序的重点则是在于以 sftp 联机上 Server 。联机的方式可以直接点选 psftp 这个档案，让他直接启动，则会出现下面的图样：

```
psftp: no hostname specified; use "open host.name" to connect
psftp>
```

这个时候可以填入您要连接上去的主机名称，例如我的区域内网络 linux.dmtsai.tw 这个主机

```
psftp: no hostname specified; use "open host.name" to connect
psftp> open test.linux.org
login as: dmtsai
Using username "dmtsai".
dmtsai@linux.dmtsai.tw's password:
Remote working directory is /home/dmtsai
psftp> <== 这里就在等待您输入 FTP 的指令了！
```

呵呵！这样就登入主机啦！很简单吧！然后其它的使用方式跟前面提到的 sftp 一样哩！加油的使用吧！

详细设定 sshd 服务器

基本上，所有的 ssh 相关设定都放在 /etc/ssh/sshd_config 里面！不过，每个 Linux distribution 的预设设定都不太相同， 所以我们有必要来了解一下整个设定值的意义为何才好！

```
[root@linux ~]# vi /etc/ssh/sshd_config
# 1. 关于 SSH Server 的整体设定，包含使用的 port 啦，以及使用的密码演算方式
# 先留意一下，在预设的档案内，只要是被批注的设定值(#)，即为『默认值！』
Port 22
# SSH 预设使用 22 这个 port，也可以使用多个 port，即重复使用 port 这个设定项目！
# 例如想要开放 sshd 在 22 与 443 ，则多加一行内容为：
# Port 443
# 这样就好了！不过，不建议修改 port number 啦！

Protocol 1,2
# 选择的 SSH 协议版本，可以是 1 也可以是 2 ，
# 如果要同时支持两者，就必须要使用 2,1 这个分隔了(Protocol 1,2)！
# 目前我们会建议您，直接使用 Protocol 2 即可！

#ListenAddress 0.0.0.0
# 监听的主机适配卡！举个例子来说，如果您有两个 IP，
# 分别是 192.168.0.100 及 192.168.2.20 ，那么只想要
# 开放 192.168.0.100 时，就可以写如同下面的样式：
ListenAddress 192.168.0.100
# 只监听来自 192.168.0.100 这个 IP 的 SSH 联机。
# 如果不使用设定的话，则预设所有接口均接受 SSH

#PidFile /var/run/sshd.pid
# 可以放置 SSHD 这个 PID 的档案！左列为默认值

#LoginGraceTime 2m
# 当使用者连上 SSH server 之后，会出现输入密码的画面，在该画面中，
# 在多久时间内没有成功连上 SSH server ，就断线！若无单位则预设时间为秒！

#Compression yes
# 是否可以使用压缩指令？当然可以啰

# 2. 说明主机的 Private Key 放置的档案，预设使用下面的档案即可！
#HostKey /etc/ssh/ssh_host_key        # SSH version 1 使用的私钥
#HostKey /etc/ssh/ssh_host_rsa_key    # SSH version 2 使用的 RSA 私钥
#HostKey /etc/ssh/ssh_host_dsa_key    # SSH version 2 使用的 DSA 私钥
# 还记得我们在主机的 SSH 联机流程里面谈到的，这里就是 Host Key ～
```

```
# 2.1 关于 version 1 的一些设定！
#KeyRegenerationInterval 1h
# 由前面联机的说明可以知道， version 1 会使用 server 的 Public Key ，
# 那么如果这个 Public Key 被偷的话，岂不完蛋？所以需要每隔一段时间
# 来重新建立一次！这里的时间为秒！不过我们通常都仅使用 version 2 ，
# 所以这个设定可以被忽略喔！

#ServerKeyBits 768
# 没错！这个就是 Server key 的长度！用默认值即可。

# 3. 关于登录文件的讯息数据放置与 daemon 的名称！
SyslogFacility AUTHPRIV
# 当有人使用 SSH 登入系统的时候，SSH 会记录信息，这个信息要记录在什么 daemon name
# 底下？预设是以 AUTH 来设定的，即是 /var/log/secure 里面！什么？忘记了！
# 回到 Linux 基础 去翻一下。其它可用的 daemon name 为：DAEMON,USER,AUTH,
# LOCAL0,LOCAL1,LOCAL2,LOCAL3,LOCAL4,LOCAL5,

#LogLevel INFO
# 登录记录的等级！嘿嘿！任何讯息！同样的，忘记了就回去参考！

# 4. 安全设定项目！极重要！
# 4.1 登入设定部分
PermitRootLogin no
# 是否允许 root 登入！预设是允许的，但是建议设定成 no！

#UserLogin no
# 在 SSH 底下本来就不接受 login 这个程序的登入！

#StrictModes yes
# 当使用者的 host key 改变之后，Server 就不接受联机，可以抵挡部分的木马程序！

#RSAAuthentication yes    # 是否使用纯的 RSA 认证！？仅针对 version 1 ！
#PubkeyAuthentication yes # 是否允许 Public Key ？当然允许啦！仅针对 version 2

#AuthorizedKeysFile      .ssh/authorized_keys
# 上面这个在设定若要使用不需要密码登入的账号时，那么那个账号的存放档案所在档名！
# 这个设定值很重要喔！档名给他记一下！

# 4.2 认证部分
#RhostsAuthentication no
# 本机系统不使用 .rhosts，因为仅使用 .rhosts 太不安全了，所以这里一定要设定为 no
```

```
#IgnoreRhosts yes
# 是否取消使用 ~/.ssh/.rhosts 来做为认证！当然是！

#RhostsRSAAuthentication no #
# 这个选项是专门给 version 1 用的，使用 rhosts 档案在 /etc/hosts.equiv
# 配合 RSA 演算方式来进行认证！不要使用啊！

#HostbasedAuthentication no
# 这个项目与上面的项目类似，不过是给 version 2 使用的！

#IgnoreUserKnownHosts no
# 是否忽略家目录内的 ~/.ssh/known_hosts 这个档案所记录的主机内容？
# 当然不要忽略，所以这里就是 no 啦！

PasswordAuthentication yes
# 密码验证当然是需要的！所以这里写 yes 啰！

#PermitEmptyPasswords no
# 若上面那一项如果设定为 yes 的话，这一项就最好设定为 no ，
# 这个项目在是否允许以空的密码登入！当然不许！

ChallengeResponseAuthentication no
# 允许任何的密码认证！所以，任何 login.conf 规定的认证方式，均可适用！
# 但目前我们比较喜欢使用 PAM 模块帮忙管理认证，因此这个选项可以设定为 no 喔！
UsePAM yes
# 利用 PAM 管理使用者认证有很多好处，可以记录与管理。
# 所以这里我们建议您使用 UsePAM 且 ChallengeResponseAuthentication 设定为 no

# 4.3 与 Kerberos 有关的参数设定！因为我们没有 Kerberos 主机，所以底下不用设定！
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosTgtPassing no

# 4.4 底下是有关在 X-Window 底下使用的相关设定！
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes

# 4.5 登入后的项目：
PrintMotd no
# 登入后是否显示出一些信息呢？例如上次登入的时间、地点等等，预设是 yes
# 亦即是打印出 /etc/motd 这个档案的内容。但是，如果为了安全，可以考虑改为 no ！
```

```
PrintLastLog yes
# 显示上次登入的信息！可以啊！预设也是 yes ！

KeepAlive yes
# 一般而言，如果设定这项目的话，那么 SSH Server 会传送 KeepAlive 的讯息给
# Client 端，以确保两者的联机正常！在这个情况下，任何一端死掉后，SSH 可以立刻知道！
# 而不会有僵尸程序的发生！

UsePrivilegeSeparation yes
# 使用者的权限设定项目！就设定为 yes 吧！

MaxStartups 10
# 同时允许几个尚未登入的联机画面？当我们连上 SSH ，但是尚未输入密码时，
# 这个时候就是我们所谓的联机画面啦！在这个联机画面中，为了保护主机，
# 所以需要设定最大值，预设最多十个联机画面，而已经建立联机的不计算在这十个当中

# 4.6 关于使用者抵挡的设定项目：
DenyUsers *
# 设定受抵挡的使用者名称，如果是全部的使用者，那就是全部挡吧！
# 若是部分使用者，可以将该账号填入！例如下列！
DenyUsers test

DenyGroups test
# 与 DenyUsers 相同！仅抵挡几个群组而已！

# 5. 关于 SFTP 服务的设定项目！
Subsystem       sftp    /usr/lib/ssh/sftp-server
```

基本上，CentOS 预设的 sshd 服务已经算是挺安全的了，不过还不够！ 建议你 (1)将 root 的登入权限取消； (2)将 ssh 版本设定为 2 。 其它的设定值就请您依照自己的喜好来设定了。通常不建议进行随便修改啦！另外，如果您修改过上面这个档案(/etc/ssh/sshd_config)，那么就必需要重新启动一次 sshd 这个 daemon 才行！亦即是：

- /etc/init.d/sshd restart

制作不用密码可立即登入的 ssh 用户：

咦！既然 SSH 可以使用 Key 来比对数据，并且提供使用者数据的加密功能， 那么可不可能利用这个 Key 就提供使用者自己进入主机，而不需要输入密码呢？ 呵呵！好主意！我们可以将 Client 产生的 Key 给他拷贝到 Server 当中，所以， 以后 Client 登入 Server 时，由于两者在 SSH 要联机的讯号传递中，就已经比对过 Key 了， 因此，可以立即进入数据传输接口中，而不需要再输入密码呢！在实作上的步骤可以是：

1. 首先，先在 Client 上面建立 Public Key 跟 Private Key 这两把钥匙，利用的指令为 ssh-keygen 这个命令；
2. 再来，将 Private Key 放在 Client 上面的家目录，亦即 $HOME/.ssh/ ， 并且修改权限为仅有该 User 可读的状态；
3. 最后，将那把 Public Key 放在任何一个您想要用来登入的主机的 Server 端的某 User 的家目录内之 .ssh/ 里面的认证档案即可完成整个程序。

说是好像很困难的样子，其实步骤真的很简单，我们依序来进行作业好了！假设前提：

- Server 部分为 linux.dmtsai.tw 这部 192.168.0.2 的主机，欲使用的 User 为 test 这个账号；
- Client 部分为 test2.dmtsai.tw 这部 192.168.0.100 PC 的 test2 这个账号， 他要用来登入 192.168.0.2 这部主机的 test 这个账号。

1. 在 Client 端建立 Public 与 Private Key ：
   建立的方法真的是简单到不行！直接在 192.168.0.100 这个 Client 上面，以 test2 这个账号，使用 ssh-keygen 这个指令来进行 Key 的产生即可！不过，需要注意的是， version 1 与 version 2 使用的密码演算方式不同，此外， version 2 提供两个密码演算的方法，我们这里仅针对 version 2 的 RSA 这个演算方法进行说明！

```
[test2@test2 ~]$ ssh-keygen -t rsa  <==这个步骤在产生 Key pair
Generating public/private rsa key pair.
Enter file in which to save the key (/home/test2/.ssh/id_rsa): <==这里按下 Enter
Enter passphrase (empty for no passphrase): <==这里按 Enter
Enter same passphrase again:  <==再按一次 Enter
Your identification has been saved in /home/test2/.ssh/id_rsa. <==这是私钥
Your public key has been saved in /home/test2/.ssh/id_rsa.pub. <==这是公钥
The key fingerprint is:
c4:ae:d9:02:d1:ba:06:5d:07:e6:92:e6:6a:c8:14:ba test2@test2.linux.org
# 注意： -t 指的是『使用何种密码演算方式？』由于我们使用 RSA ，
# 所以直接输入 -t rsa 即可建立两支 Keys ！
# 此外，建立的两把 Keys 都放置在家目录下的 .ssh 这个目录中！
# 察看一下这两把 Keys 吧！

[test2@test2 ~]$ ll ~/.ssh
total 12
-rw-------    1 test2    test2         887 Nov 12 22:36 id_rsa
-rw-r--r--    1 test2    test2         233 Nov 12 22:36 id_rsa.pub
-rw-r--r--    1 test2    test2         222 Oct 31 11:20 known_hosts
```

   请注意上面喔，我的身份是 test2 ，所以当我执行 ssh-keygen 时， 才会在我的家目录底下的 .ssh/ 这个目录里面产生所需要的两把 Keys ，分别是私钥(id_rsa)与公钥(id_rsa.pub)。另外一个要特别注意的就是那个 id_rsa 的档案权限啦！他必须要是 -rw------- 才好！否则内容

被人家知道了，那么您的 Keys 不就有可能外泄了？所以请特别留意他的权限喔！ 那么那个 id_rsa.pub 则是『公钥！』这个档案必须要被放置到 Server 端才行！

2. 在 Client 端放置私钥：
在预设的条件中，我们的私钥必需要放置在家目录底下的 .ssh 里面，那么如果是 version 2 的 RSA 算法，就需要放置在 $HOME/.ssh/id_rsa 当中！咦！刚好使用 ssh-keygen 就是已经产生在这个目录下了，所以自然就不需要去调整他了！以我的 test2.dmtsai.tw 来看，那么我的档案就会放置在 /home/test2/.ssh/id_rsa 这个档案就是私钥啦！

3. 在 Server 端放置可以登入的公钥：
既然我们要让 test2 可以用 test 这个账号登入 linux.dmtsai.tw 这部主机，那么这部主机自然需要保有 test2 的 public key 啰！对的！所以我们必需要将 Client 端建立的 id_rsa.pub 档案给他拷贝到 linux.dmtsai.tw 里头的 test 这个使用者的家目录之下！那么如果您还记得上面的 sshd_config 这个档案的设定的话，那么应该就记得『 AuthorizedKeysFile 』这个设定吧！是的！ 在被登入的主机的某个账号，他的公钥放置的文件名称预设就是这个项目所记载的！而他预设的档名就是 authorized_keys 这个文件名称啦！那么应该怎么做呢？

```
1. 先在 Client 端以 sftp 将公钥丢到 test 上面去！
[test2@test2 ~]$ cd ~/.ssh
[test2@test2 .ssh]$ scp id_rsa.pub test@192.168.0.2:~/
test@192.168.0.2's password:
id_rsa.pub          100%   233     0.2KB/s    00:00

2. 到 Server 上面，将公钥转存到 authorized_keys 档案中！
[test@linux ~]$ cd ~/.ssh
[test@linux .ssh]$ cat ../id_rsa.pub >> authorized_keys
```

请注意上面的机器！由于 authorized_keys 可以保存相当多的公钥内容，因此， 可以使用 >> 的方式来将 Client 端的公钥新增到该档案内！呵呵！做完这一步一后，未来 test2 就可以直接在 test2.dmtsai.tw 以

```
[test2@test2 ~]$ ssh test@linux.dmtsai.tw
```

这样就可以不需要输入密码啰！但是请注意， test 不能以 test2 登入 test2.linux.org 喔！

很简单的步骤吧！这样一来，就可以不需密码的手续了！无论如何，您要记得的是：

- Client 必须制作出 Public & Private 这两把 keys，且 Private 需放到 ~/.ssh/ 内；
- Server 必须要有 Public Key ，且放置到使用者家目录下的 ~/.ssh/authorized_keys；

未来，当您还想要登入其它的主机时，只要将您的 public key (就是 id_rsa.pub 这个档案) 给他 copy 到其它主机上面去，并且新增到某账号的 ~/.ssh/authorized_keys 这个档案中！哈哈！成功！

安全设定：

老实说，大家都被『SSH 是个安全的服务』所欺骗了！其实 sshd 并不怎么安全的！翻开 openssh 的过去历史来看，确实有很多人是利用 ssh 的程序漏洞来取得远程主机 root 的权限，进一步黑掉对方的主机！

sshd 之所谓的『安全』其实指的是『 sshd 的数据是加密过的，所以他的数据在 Internet 上面传递时是比较安全的。 至于 sshd 这个服务本身就不是那样安全了！所以说：『非必要，不要将 sshd 对 Internet 开放可登入的权限，尽量局限在几个小范围内的 IP 或主机名称即可！ 这很重要的喔！

好了，那么关于安全的设定方面，有没有什么值得注意的呢？当然是有啦！ 我们可以先建议几个项目吧！分别可以由：

- /etc/ssh/sshd_config
- /etc/hosts.allow, /etc/hosts.deny
- iptables

这三方面来着手进行！底下我们就说一说吧！

- /etc/ssh/sshd_config

一般而言，这个档案的预设项目就已经很完备了！所以，事实上是不太需要更动他的！ 但是，如果您有些使用者方面的顾虑，那么可以这样修正一些问题呢！

- 禁止 root 的登入：
  任何时候，不许 root 以远程联机的方式登入，都会是一个好主意！所以这里蛮建议大家直接将 root 的登入权限拿掉吧！所以，可以修改 /etc/ssh/sshd_config 这个档案的内容为：

  ```
  [root@linux ~]# vi /etc/ssh/sshd_config
  PermitRootLogin no     <== 将他改成 no 吧!
  [root@linux ~]# /etc/init.d/sshd restart
  ```

  如此一来，以后 root 就不能以 ssh 登入啰！这样还是比较好的啦！ ^_^

- 不许某个群组登入：
  有些特殊情况中，我们想要让使用者只能使用 sendmail, pop3, ftp 等，但是不希望他可以远程联机进来，那么您可以这样做：

  1. 将这些使用者都归纳在某一个特殊群组之下，例如 nossh 这个群组好了；
  2. 在 /etc/ssh/sshd_config 当中加入这一行：『 DenyGroups   nossh 』
  3. 重新启动 sshd ： /etc/init.d/sshd restart

  这样就 OK 啦！

- 不许某个使用者登入：
  跟 DenyGroups 类似，使用 DenyUsers 即可！参考 sshd_config 的设定喔！

* /etc/hosts.allow 及 /etc/hosts.deny

简单的方法就是：

```
[root@linux ~]# vi /etc/hosts.allow
sshd: 192.168.0.1, 192.168.0.2, 192.168.0.3, 192.168.0.4, 192.168.0.5: allow

[root@linux ~]# vi /etc/hosts.deny
sshd : ALL : spawn (/bin/echo Security notice from host `/bin/hostname`; \
/bin/echo; /usr/sbin/safe_finger @%h ) | \
/bin/mail -s "%d -%h security" root@localhost & \
: twist ( /bin/echo -e "\n\nWARNING connectin not allowed.". )
```

* iptables

多几层保护也很好的！所以也可以使用 iptables 喔！参考： 简易防火墙架设 一文啰！

最后，『鸟哥呼吁大家，不要开放 SSH 的登入权限给所有 Internet 上面的主机～』 这很重要喔～因为如果对方可以 ssh 进入您的主机，那么......太危险了～

XDMCP 服务器

考虑一个情况，如果您的 Linux 主机上面主要是用来作为图形处理时，而且同时有多人需要用到那个功能，那么一部 Linux 是否一次仅能提供一个人处理那个软件呢？嘿嘿！那可不一定喔！因为 Linux 有相当优秀的 X Window System 啊！

X Window 的 Server/Client 架构

X Window System 的架构对于常常玩网络的朋友来说(这也包括鸟哥啦！ @_@)实在不太好理解～ 因为 X Window System 在运作的过程中，同样包含了 X Server 与 X Client 这两个东西， 但是他的作用却与网络主机的 Server/Client 架构大异其趣喔～先来说说 X Server/Client 所负责的东西：

* X Server： 他主要负责的是屏幕画面的绘制与显示。 X Server 可以接收来自 X client 的数据，将这些数据绘制呈现为图面在屏幕上。此外，我们移动鼠标、点击数据、由键盘输入数据等等， 也会透过 X Server 来传达到 X Client 端，而由 X Client 来加以运算；
* X Client： 他主要负责的是数据的运算。 X Client 在接受到 X Server 传来的数据后，会经由本身的运算，而得到鼠标应该要如何移动、 点击的结果应该要出现什么样的数据、键盘输入的结果应该要如何呈现等等， 然后将这些结果告知 X Server ，让他自行去绘制到屏幕上。

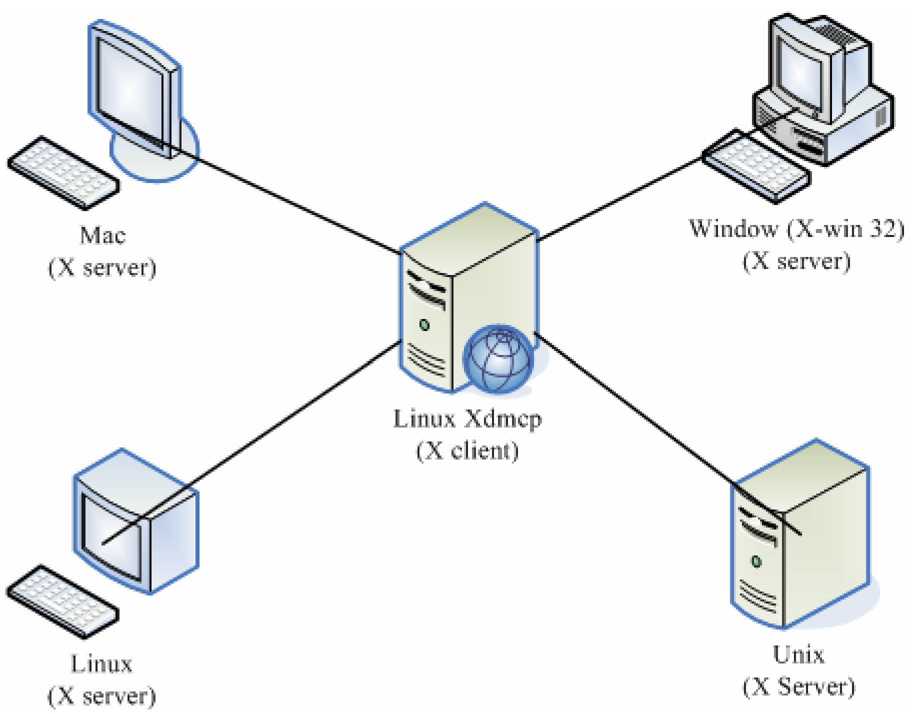
这样说可以理解吗？也就是说，我们移动鼠标或敲打键盘时， X server 可以接受到这些硬件所输入的数据， 但他不知道应该要怎么作才好，就把这些数据告诉 X Client ，此时， X Client 就会将这些数据计算， 最后得到鼠标应该要如何移动与键盘的数据应该要如何呈现，并将这些结果告知 X Server ， 而 X Server 就会经由 X Client 的告知，而将那些数据数据在屏幕上面呈现出来。

事实上，X Server 与 X Client 通常是在同一部机器上面的，例如我们在 Linux 上面执行有名的 KDE 这个桌面一样。 但是 X Server/Client 却不必然一定要在同一部机器上的， 也就是说，我们可以透过网络连接两部主机的 X 系统呢！ 这也是最早 X 系统开发时的概念。不过，这个时候的 X server 指的是哪一部呢？

举个例子来说，我们可以在 Windows 系统上面执行一套 X Win32 的软件， 他可以连接到 Linux 的 X 上头而让使用者以图形接口登入 Linux 。 此时，因为『X Win 32』主要是在屏幕上面显示，他的做用就是屏幕绘制，因此他是 X Server 。 不信的话，等一下我们测试底下的资料时，您会发现，启动 X win32 这套软件后， 在 Windows 系统上就会出现 port 6000 这个 X server 的 port 呢～而这个时候 X Win 32 软件其实就是连接到 Linux 的某个程序，我们等一下要介绍的 XDMCP 就是其中一种。这个 XDMCP 可以将 X Win32 传送过来的数据运算成可以绘制的数据而回传给 X win 32 那套软件， 此时的 XDMCP 程序就是一个 X Client 啰～

所以啦！如果您的 Windows 想要连接到 Linux 主机的话，那么 Windows 就得要有可以执行的 X Server 软件啦～ 而 Linux 主机则必须要启动一个可以接受 X Server 资料运算的 X Client 啦～ 就是这样说～

但是，这样做有什么好处呢？呵呵～最大的好处就是，在服务器上的 X Client 不需要知道 X Server 的硬件是什么～因为负责显示的是 X server 的事情，管理硬件的动作也是 X server 在做， 在主机上的 X Client 只是将这些鼠标移动与点击还有键盘的输入等的数据在主机端运算后， 最后将结果传送给 X Server 显示而已。(当然啦， X Client 的运算内容还是会用到主机端的设定文件与函式库就是了。)



图十、X server/client 的架构

那么什么时候会出现多使用者连入 X 主机的情况呢？以鸟哥的例子来说，我们实验室有一部 Linux 在进行数值模拟， 他输出的结果是 NetCDF 档案，我们必须使用 PAVE 这一套软件去处理这些数据， 以绘制等浓度图等等的。但是我们有两三个人同时都会使用到那个功能， 偏偏 Linux 主机是放在机架柜里面的，要我们挤在那个小小的空间前面『站着』操作计算机， 可真是讨人厌啊～这个时候，我们就会架设图形接口的远程登入服务器， 让我们可以『多人同时以图形接口登入 Linux 主机』来操作我们自己的程序！ 很棒，不是吗？！

设定 XDMCP

XDM 是 X Display Manager 的简称，他的功能是什么呢？ 简单的说，就是管理操控 X Server 的显示啦～ 他主要有两种管理方式， 如果 X Server/Client 在同一部机器上，那么启动 xdm 之后，就会产生一个 X server 了； 而如果 X server/client 不在同一部主机上面，那么启动 xdm 后，他就会透过网络去管理远程那部主机的 X server 了。而 XDMCP (X Display Manager Control Protocol) 就是负责监听来自网络上面对于 xdm 的要求的啦～

由 X11 (CentOS 使用的是 Xorg 这个计划的 X11)提供的 display manager 为 xdm ，设定档在 /etc/X11/xdm/xdm-config ， 而著名的 KDE 与 GNOME 也都有自己的 display manager 管理程序，分别是 kdm 与 gdm ， 设定档则是 /etc/X11/xdm/kdmrc 与 /etc/X11/gdm/gdm.conf (不同的 distribution 这个档案放置的目录不太一样)。我们可以透过三者中任何一者的 display manager 的设定档来启动 xdmcp 这个协议呢～

要启用 xdmcp 的功能真的很简单，如果您要启用 xdm 的话，修改 /etc/X11/xdm/xdm-config 这个档案，找到底下这一行(一般在最后一行)：

```
DisplayManager.requestPort:    0
```

将他修改成为：

```
!DisplayManager.requestPort:    0
```

亦即是批注掉，然后再重新启动 xdm 就好了。而 kdm 与 gdm 的设定也类似，底下鸟哥主要以 kdm 来进行 xdmcp 的架设。不过要注意的是，即使在 Linux 主机端不启用 X Server (port 6000) 也是可以正确无误的提供 X 接口的登入的～就如同上面提到的概念一般～ 但是，如果要获得比较正确的讯息，那么还是建议您，启用 kdm 时一并正确的启动 X ， 只是安全性上面就要注意一些了！好了，多说无益，来实作吧！

```
1. 先让 kdm 支持 xdmcp 模式
[root@linux ~]# cd /etc/X11/xdm
[root@linux xdm]# vi kdmrc
[Xdmcp]
Enable=1
# 大约是在 70 行左右。不要怀疑！真的只要这样就好了！

2. 让 client 可以透过 X 来登入系统！与权限有关的设定
[root@linux xdm]# vi Xaccess
*
# 为了安全性上面的需要，想要登入 X 的话，得要通过这个档案的验证才行。
# 找到上面这一行，如果没有这一行的话(整行只有一个 * )，
# 就自行加入。这表示『不论来自哪里，我都接受 X 登入』的意思！

3. 启动 kdm 喔！
[root@linux xdm]# /etc/init.d/xfs start
# 就如同我们上面提到的， kdm 执行后，可能的话，会在本机端启动一个 X server 的，
# 而我们这一版的 Xorg 要顺利的启动，得要先启用 X font Server 才行，
```

```
# 否则的话，您就得要到 /etc/X11/Xorg.conf 里面去设定好每个字型的路径才行。
[root@linux xdm]# kdm
[root@linux xdm]# netstat -tlunp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address   State  PID/Program name
tcp        0      0 0.0.0.0:6000    0.0.0.0:*         LISTEN 5920/X
tcp        0      0 :::6000         :::*              LISTEN 5920/X
udp        0      0 :::177          :::*                     5918/kdm
# 要看到有 177 的 udp port 出现才行～ 因为那是 xdmcp 协议的监听埠口。
# 不过，如果要看看是否有成功启动 X 的话，就得要查阅 6000 这个 port 啰～
# 如果没有看到 port 6000 的话，请查阅 /var/log/Xorg.0.log 喔！
# 如果想要设定开机就自动执行的话，可以利用 chkconfig 加入 xfs ，
# 也可以将 kdm 这个指令写到 /etc/rc.d/rc.local 这个档案中～
```

虽然是非必备的，不过为了避免困扰，这里还是得要提醒大家。(因为 CentOS 不需要启动 X 就能够提供 xdmcp 登入) 鸟哥上面的测试是在 run level 为 3 的环境下，且整体在执行的时候， /var/log/messages 与 /var/log/Xorg.0.log 这两个档案内容中并没有 kdm 的相关错误讯息～ 很重要啊！因为某些套件如果没有成功的启动 X 时，他就无法提供登入呢～

客户端登入

- 客户端是 Linux 主机：

如果想要进行 XDMCP 提供的 X 接口的登入 Linux 主机时，在 Linux 底下可是容易的很～ 底下的流程是在『客户端』执行的喔～不是刚刚那部 XDMCP 所在的 Linux 主机啦！

```
0. 请务必要在 X Window 当中，进入 X Window 的方式有：
[root@client ~]# startx
# 或
[root@client ~]# init 5

1. 在 X Window 的画面当中，启用一个 shell ，然后输入：
[root@client ~]# xhost + 192.168.1.100
192.168.1.100 being added to access control list
# 假设我刚刚那部 Linux 主机的 IP 为 192.168.1.100
[root@client ~]# init 3    <== 关闭 X Server

2. 在文字接口下输入：
[root@client ~]# X -query 192.168.1.100
# 进入 X Window 啰！
```

如果一切顺利的话，您应该就能够到 X Window 的画面底下去登入远程主机啰～

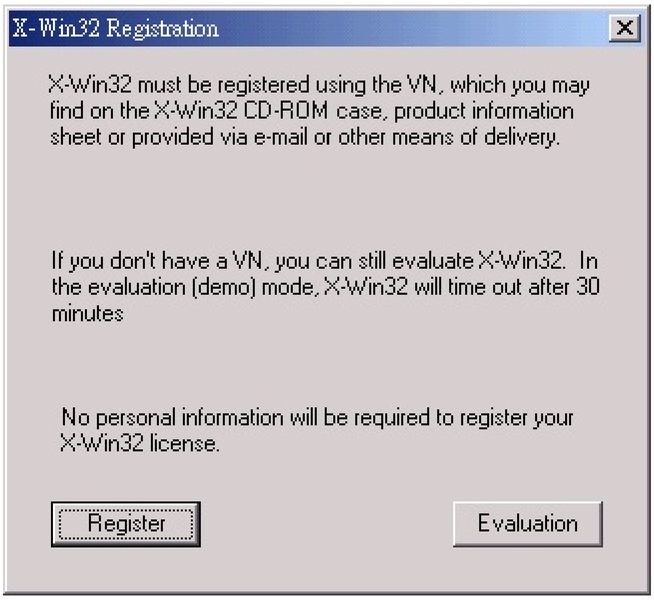
- 客户端是 Windows 主机：

如果想要进行 XDMCP 提供的 X 接口的登入 Linux 主机时，在 Windows 底下就得要使用其它软件来支持了。例如：

- X-Win32 (http://www.starnet.com/evalkey/)
- Exceed (http://www.hummingbird.com/products/nc/exceed/index.html?cks=y)

这里鸟哥用 X-Win32 来进行测试。正个运作流程是这样的：

1. 安装 X-Win32 ，很简单～就是直接执行下一步即可。比较可惜的是，这个软件目前没有中文支持喔！
2. 直接在『开始』-->『程序集』-->『X Win 32』执行『X-Win32』这支程序～会出现如下图：



图十、X Win 32 执行范例

上面只是在告诉我们,这个软件需要注册。但是我们只是试用而已,所以可以直接按下 Evaluation 即可。 不过，试用版有联机三十分钟的限制就是了～@_@

3. 在出现的窗口当中，当然要选择 XDMCP 这个模式啰～

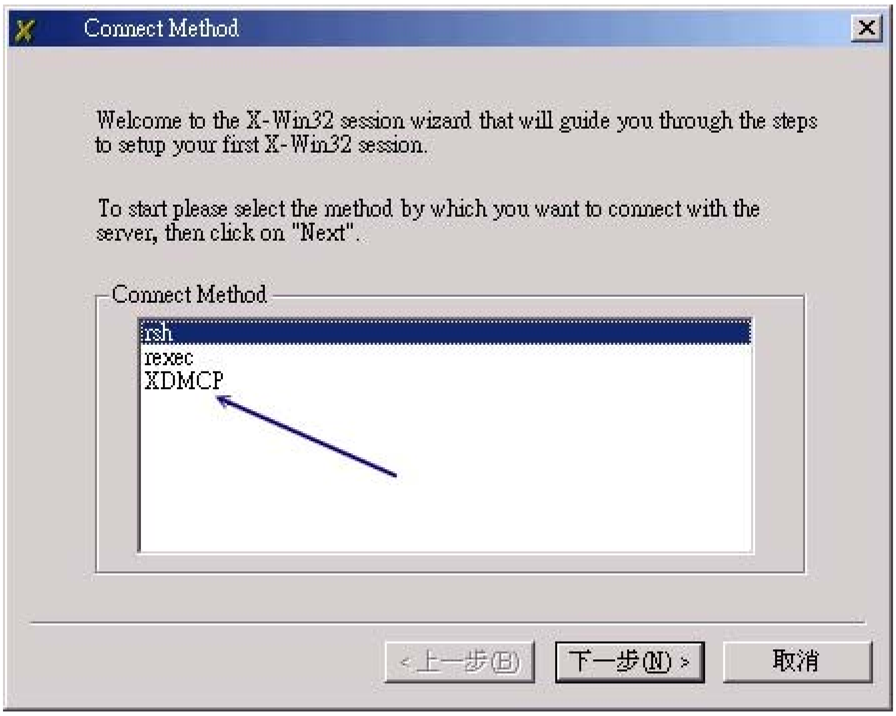


图 11 、X Win 32 执行范例

4. 因为我们可以直接给予一个 X server 的 IP，所以这里我们可以选择『Query』这个项目。 如果您是在 LAN 环境当中，而且 Client/Server 是在同一个网段时，其实可以选择 Broadcast 比较好用！无论如何，这里鸟哥先以 Query 来介绍。

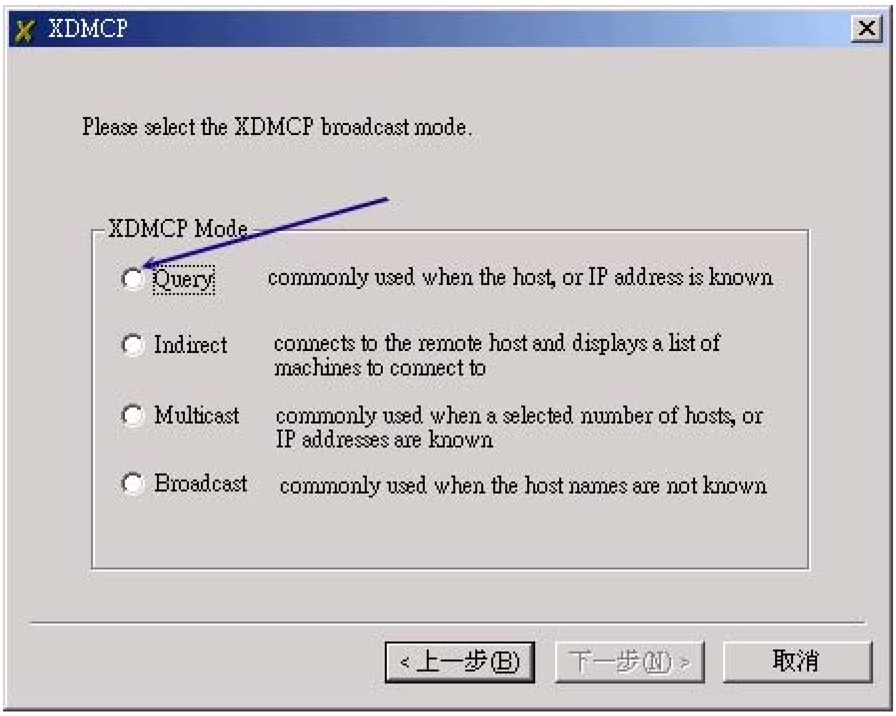


图 12 、X Win 32 执行范例

5. 接下来的画面可以填入 IP 或主机名称啊！建议直接输入 IP 啦！



图 13 、X Win 32 执行范例

6. 接下来的画面只要填入一个简单的好记得名称即可！如果想要立即执行的话，那个『Launch this session now』 可以直接打勾喔！

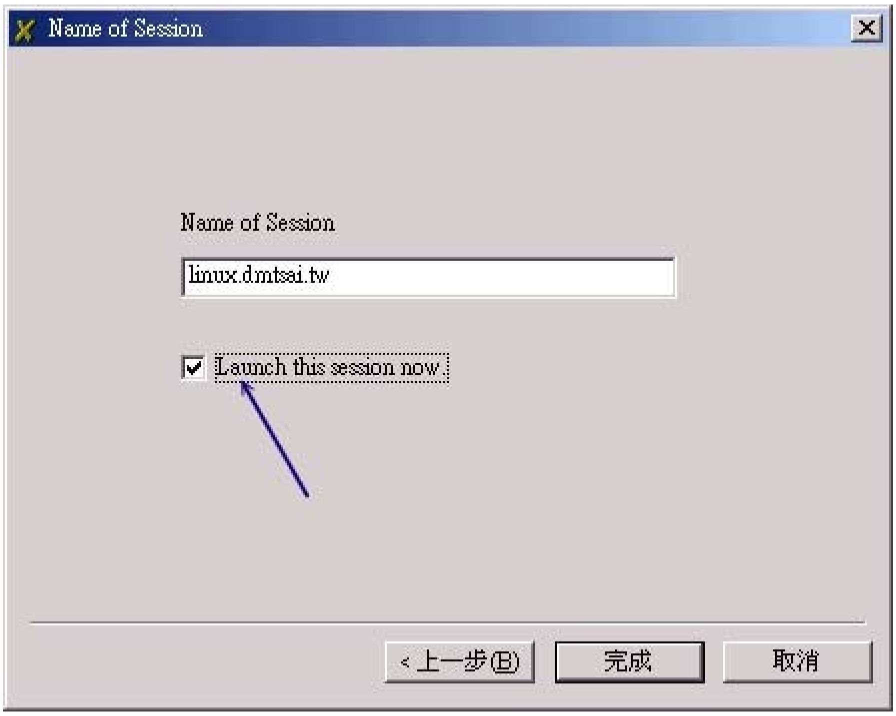


图 14 、X Win 32 执行范例

7. 理论上，这样应该就可以立即的进行联机到 X Window Server 才对。不过，如果没有成功呢？ 没关系！我们可以重新来修改一下设定啊～如果执行了 X-Win32 之后，在工作列的右下角会出现 X 图示， 如下所示：

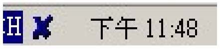


图 15 、X Win 32 执行范例

将鼠标指针移动到 X 上头，按下右键，可以得到如下的选单出现：



图 16 、X Win 32 执行范例

在上图上面按下『XConfig』就可以出现底下的图示：

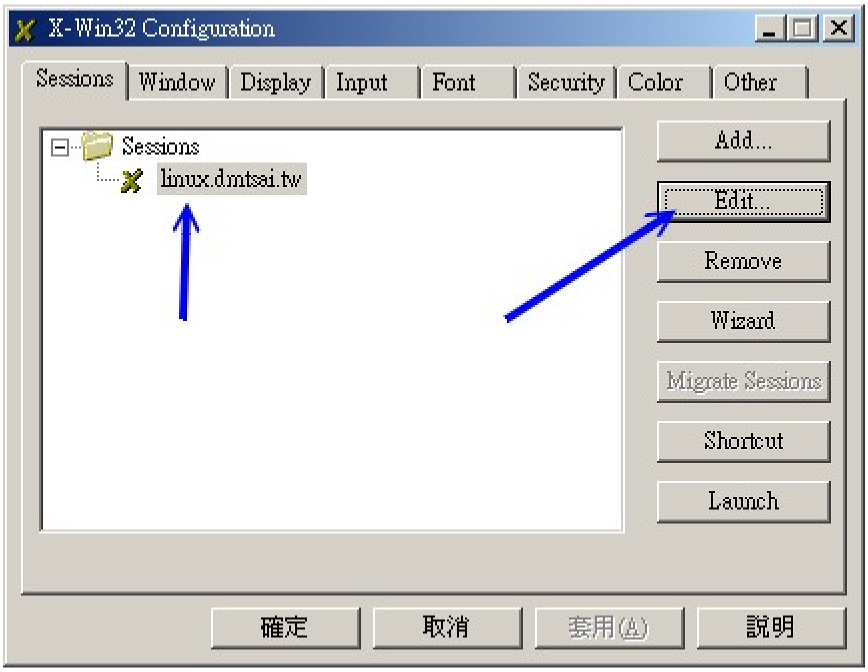


图 17 、X Win 32 执行范例

然后选择我们刚刚设定好的那个 session ，按下『Edit』，就可以开始修改刚刚的设定值啰～ 更多的选项请自行参考 X Win32 当中的说明。此时，我们可以在工作列的 X 上面，按下左键， 应该会出现所有可以用的 session ，请选择 linux.dmtsai.tw 那个 session ，如果一切顺利， 就会出现如下的画面：



图 18 、X Win 32 执行范例

输入账号密码之后，嘿嘿！立刻就可以在 Windows 上面看到您 Linux 主机的 X Window 画面了～感动吧～ ^_^

Tips:
事实上，xdmcp 真的很容易设定的～鸟哥曾经以 gdm (设定档为 gdm.conf) 及 kdm 分别设定过， 执行上都没有问题。不过，需要特别留意的是，因为 X Window 执行的数据量实在是太大了， 所以，如果您在 Internet 上面使用 ADSL 传输的话，想要玩这个玩意儿～ 奉劝您：『别想了～』这东西主要还是应用在内部网域当中的啦！



关闭 XDMCP

如果想要关掉的话，就这样做：

```
[root@linux xdm]# killall -9 kdm
[root@linux xdm]# /etc/init.d/xfs stop
```

这样就可以将 xdmcp 给他关掉啰～ ^_^

VNC 服务器

虽然 xdmcp 就已经很好用了，不过，就以传输速度上来讲，他真的是慢啊～～ 这个时候，我们可以利用 VNC (Virtual Network Computing) 这个好用的咚咚来进一步设定我们的 X Window 登入系统喔。

VNC 必须要透过 VNC Server 与 VNC client 软件的呼相搭配，就可以进行比较快速一点的数据传输。 而 VNC 如果想要漂亮的一点的话，也是需要搭配 xdmcp 的啦～因为如果是纯粹使用 VNC 连接到 Xorg (或 XFree86) 那个简单的画面，真的是....有点不好用～

其实 VNC Server 会在主机多开一个程序在等待 Client 的登入要求， 等到 Client 登入之后，才去执行 Window manager 的启动。而这个 Window manager 的启动方式有很多种， 最阳春的就是利用 Xorg 预设的 twm 这个窗口管理程序，他真的是不好看～ 画面有点像这样：

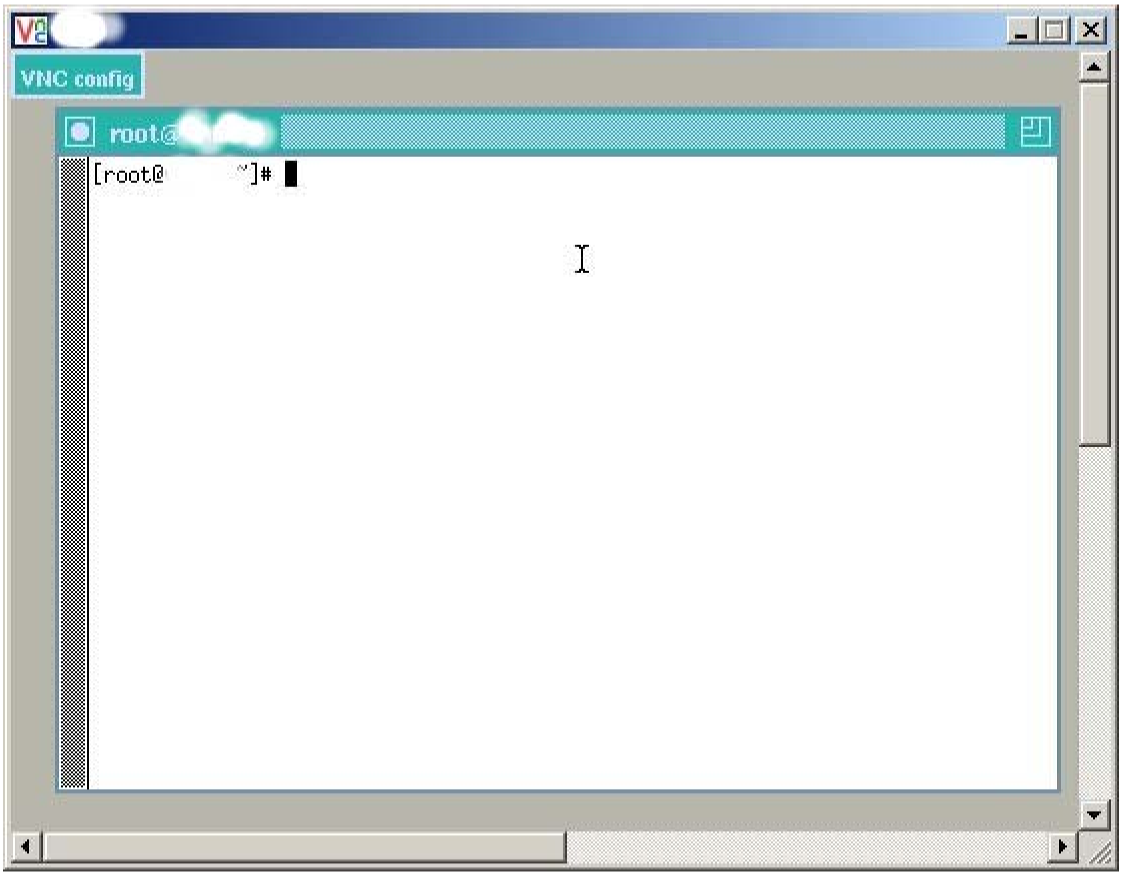


图 19 、使用 twm 联机为 VNC Server 的执行范例

真的不很好看喔～那怎么办？其实我们可以透过更改 VNC 的启动设定档： xstartup 来设定不同的 Window manager ， 另外，我们也可以透过启用 kdm 或 gdm 这两个好用的 display manager 来代为管理 Window manager 呢～ 鸟哥比较喜欢使用查询 (Query) XDMCP 的方式来启动 VNC ，而不是直接启动 startkde 这个程序的说～ 所以，底下我们就直接来设定可以连接到 xdmcp 上的 VNC Server 吧！

```
1. 先让 kdm 支持 xdmcp 模式
[root@linux ~]# cd /etc/X11/xdm
[root@linux xdm]# vi kdmrc
```

```
[Xdmcp]
Enable=1

2. 让 client 可以透过 X 来登入系统！与权限有关的设定
[root@linux xdm]# vi Xaccess
*

3. 启动 kdm 喔！
[root@linux xdm]# /etc/init.d/xfs start
[root@linux xdm]# kdm
[root@linux xdm]# netstat -tlunp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address   State  PID/Program name
tcp        0      0 0.0.0.0:6000    0.0.0.0:*         LISTEN 5920/X
tcp        0      0 :::6000         :::*              LISTEN 5920/X
udp        0      0 :::177          :::*                     5918/kdm
# 要看到有 177 的 udp port 与 port 6000 才行；
# 如果没有看到的话，就得要查询底下几个档案的内容，看看错误讯息了！
# a. 必须查阅 netstat -tlunp
# b. 必须查阅 /var/log/Xorg.log.0
# c. 必须查阅 /var/log/messages
# d. 必须查阅 /var/log/kdm.log

4. 用某身份建立 passfile 给 VNC 联机时使用
# 因为 VNC 开的每个 port 都是给某特定使用者登入的，因此，
# 每个 VNC server 都会启用自己的 port 呢～据说最大可开放到 10 个～
# 鸟哥这里假设利用 dmtsai 这个使用者来执行 VNC ，那么他就必须要有底下几个动作：
4.1 建立联机用密码
[root@linux xdm]# su dmtsai
[dmtsai@linux xdm]$ vncpasswd
Password: <== 这里请输入密码
Verify: <== 再输入一次～
# 特别注意，为了安全起见，密码的长度是有限制的！
# 至少要大于六个字符，且不能与账号相同～
# 密码建立后，会在 /home/dmtsai/.vnc/passwd 这个档案中记录了你的密码～
# 同时，在这个目录下，还有设定档 xstartup 可以利用喔！ ^_^
4.2 修改设定档 xstartup
[dmtsai@linux xdm]$ vi /home/dmtsai/.vnc/xstartup
# 将这个档案内的所有数据通通给他批注掉～不需要保留～
4.3 离开此一身份使用者的画面
[dmtsai@linux xdm]$ exit

5. 修改 /etc/sysconfig/vncserver 档案内容
```

```
# 这个档案是 FC4 预设的启动 VNC 的读取档，所以我们可以修改他～
[root@linux xdm]# vi /etc/sysconfig/vncservers
# 将原本的数据改成这样：
VNCSERVERS="2:dmtsai"
VNCSERVERARGS[2]="-geometry 800x600 -query localhost"
# 意思是说，我们要启动一个 VNC 在 port 5900+2 即 5902 的意思，

6. 启动 VNC server
[root@linux xdm]# /etc/init.d/vncserver start
# 此时在 /home/dmtsai/.vnc/ 里面应该会有几个档案您应该要注意的，
# 最重要的就是 dmtasi.linux.dmtsai.tw:2.log 这个档案，档名的由来是：
# username.hostname.domainname:[port number].log ，因为我们是启用 5902 ，
# 所以就有 :2.log 的附档名啦～务必看到里面没有错误才行喔～
# 如果发现找不到/usr/X11R6/lib/X11/xserver/SecurityPolicy 的错误，先略过不要紧～

7. 查阅设定结果
[root@linux xdm]# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State    PID/Program name
tcp        0      0 0.0.0.0:5802  0.0.0.0:*       LISTEN   15287/Xvnc
tcp        0      0 0.0.0.0:5902  0.0.0.0:*       LISTEN   15287/Xvnc
tcp        0      0 0.0.0.0:6000  0.0.0.0:*       LISTEN   15019/X
tcp        0      0 0.0.0.0:6002  0.0.0.0:*       LISTEN   15287/Xvnc
tcp        0      0 :::6000       :::*            LISTEN   15019/X
tcp        0      0 :::6002       :::*            LISTEN   15287/Xvnc
udp        0      0 0.0.0.0:32924 0.0.0.0:*                15287/Xvnc
udp        0      0 :::177        :::*                     15017/kdm
```

设定好像也很简单喔～那么鸟哥干嘛讲这么多原理？ 原因无他，因为希望大家可以在不同的 Linux distributions 也能够顺利的架设好 XDMCP 与 VNC ，如果您能够善用登录文件的内容信息， 那么应该会比较容易 debug 的啦～ ^_^

另外，事实上启动 VNC 的 script 是由 vncserver 这个指令所启用的， 您也可以直接利用某个身份直接下达：

```
[root@linux ~]# vncserver :3

You will require a password to access your desktops.

Password: <== 就输入密码吧！
Verify: <== 再输入密码吧！

New 'dmtsai.linux.dmtsai.tw:3 (dmtsai)' desktop is dmtsai.linux.dmtsai.tw:3

Starting applications specified in /root/.vnc/xstartup
```

```
Log file is /root/.vnc/dmtsai.linux.dmtsai.tw:3.log
```

如此一来，就可以启用一个 port 为 5903 的 VNC 服务啰～ 您可以再度的去到 logfile 查一查啊～至于关闭的话，可以用：

```
[root@linux ~]# vncserver -kill :3
```

这样就能够关闭啰～呵呵！那么如果想要连接到 VNC Server 的话， 在 Linux 底下可以利用 KDE 的 krdc 这支远程联机程序，如果是 Windows 的话， 就得需要 VNC Client 啰～您可以前往底下的网站：

- http://www.realvnc.com/download.html

直接下载 Free Edition 来测试看看就好了。安装过程我们就不提了～ 安装完毕之后，直接执行『开始』 -->『程序集』-->『RealVNC』-->『Run VNC viewer』后， 出现如下的窗口：

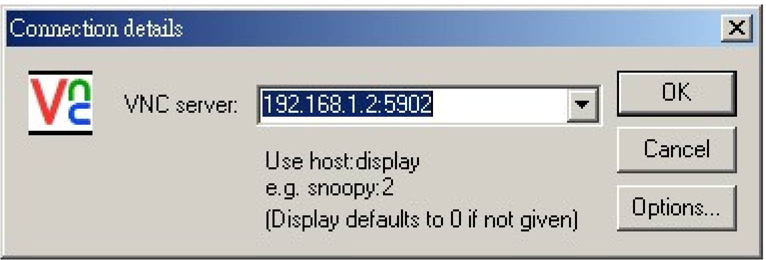


图 19 、VNC viewer 执行范例

输入了您的主机 IP 与该 VNC 对应的 port 之后，会出现一个密码窗口：



图 20 、VNC viewer 执行范例

记得啊～这里的密码指的是『您利用 vncpasswd 所建立的密码』，而不是登入者的密码啊～ 按下 Enter 之后，如果顺利的话，就会出现如下的图示啰～

很不错吧！ ^_^ 这样就能够在 Client 端登入 Linux 主机啰～ 而且还可以多人共享呢～真是棒～ ^_^。

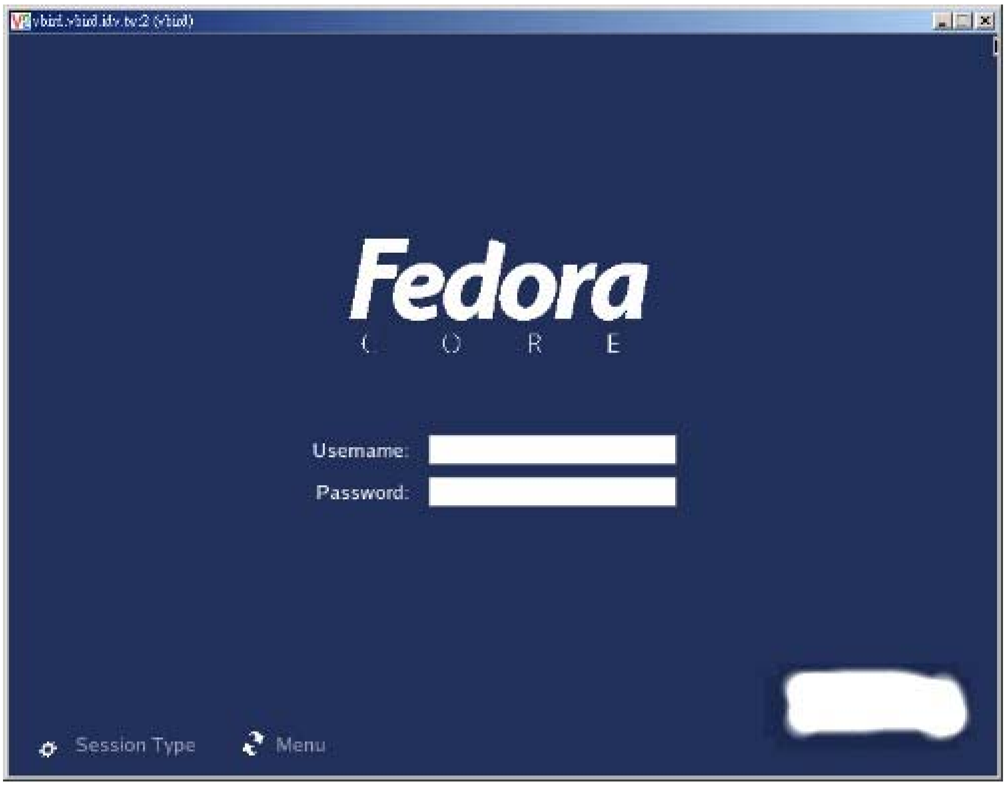


图 21 、VNC viewer 执行范例



图 22 、VNC viewer 执行范例

很不错吧！ ^_^ 这样就能够在 Client 端登入 Linux 主机啰～ 而且还可以多人共享呢～真是棒～ ^_^。但是，如果您设定完毕之后， 在登录档老是出现这个咚咚：『XDMCP fatal error: Manager unwilling Host

unwilling』， 就是 /etc/X11/xdm/Xaccess 这个档案的设定需要变更了！

另外，有些朋友一定会觉得奇怪，那就是，为甚么我的 VNC 服务器的 server / client 端画面并不是同步的呢？这是因为 Linux 本身提供多个 VNC server ，她们是各自独立的，所以当然就不会与 tty7 的画面同步了。 但是如果您想要与 Linux 的 tty7 同步的话，可以利用 VNC 释出的给 X Server 使用的模块来加以设定即可。 如果您是 CentOS 4.x 这个 distribution 的话，恭喜您，系统预设已经将 vnc.so 这个模块释出了，您可以查阅 /usr//X11R6/lib/modules/extensions/ 这个档案，即可知道有没有 vnc.so 这个模块。如果您没有这个模块的话， 请参考 http://phorum.study-area.org/viewtopic.php?t=25713 这一篇文章的说明， 依序来设定吧！

```
[root@linux ~]# vi /etc/X11/xorg.conf (或 XF86Config)
Section "Module"
          ....
        Load  "vnc"
EndSection
# 在 Module 这个 section 当中加入 vnc 这个模块即可
Section "Screen"
        Identifier "Screen0"
        Device     "Videocard0"
        Monitor    "Monitor0"
        Option     "passwordFile"  "/etc/vnc/passwd"
        DefaultDepth     16
          ......
EndSection
# 假设您的 vnc 密码档案放置在 /etc/vnc/passwd 里头，
# 这个时候就得要将密码文件内容写到 Screen 这个 section 当中了
```

此时给他重新启动一下 kdm 或者是重新进入 run level 5 的时候，您就会发现多了一个 port 5900 呢，嘿嘿，准备同步登入吧 ^_^

RSH 服务器

什么是 RSH 服务器呢？其实，这是早期的不同主机之间互相『直接操作』对方资源的一个方法。 其实就好像使用『 ssh dmtsai@localhost date 』之类的执行方法啦！ 我们可以透过 rsh 来操作对方主机啊。这个 RSH 就是被称为 R Shell 的咚咚啰～

目前 RSH 很少被使用到一般的服务器上面，尤其是对 Internet 开放的主机， 这是因为 RSH 的危险性很高！他不但是明码传输，而且一个设定不良，可能会让所有人都能使用 RSH 来登入主机！ 不过，RSH 却是操作丛集计算机 (cluster) 里面最常见的服务之一！

所谓的丛集计算机，简单的说就是『将很多部主机透过网络连结在一起， 以其中一部主机作为主要操控计算机 (或者称为 master) ，其它主机仅负责来自 master 的要求 (所以被称为 slave)；所有的计算机工作都是由 master 所掌控， slave 仅负责运算的部分。』 这也就是说， slave 大概就仅提供 CPU 的运算单元，其它的事都是 master 负责来运作。 当然，我们也是只要操控 master 那部主要计算机而已。在

这样的情况下， slave 接受 master 的指令， 最主要就是透过 RSH 啊！ (当然，也可以透过 SSH 配合金钥来达成这样的工作！)。 关于更多的 cluster 的介绍，可以参考一下鸟哥之前写的一篇小文章：

http://linux.vbird.org/linux_server/0600cluster.php

至于 RSH server 与 Client 的互动可以使用下面的图示来查阅：

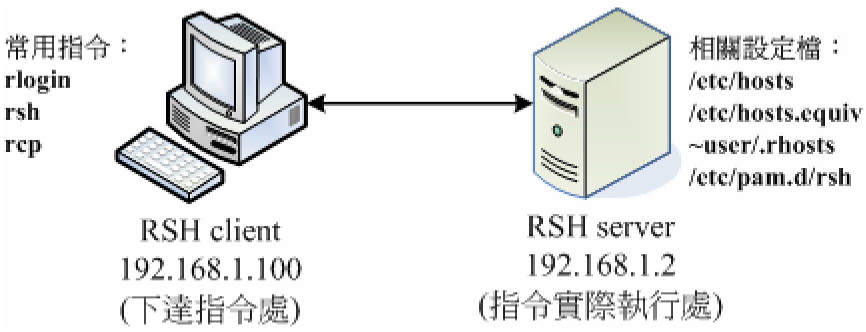


图 23 、RHS Server/Client 互动示意图

上图中在 RSH server 当中的几个设定档是这样的：

- /etc/hosts：主要规范 RSH server/client 的主机名称与 IP 对应！
- /etc/hosts.equiv：规范出哪一部 client 可以连上这部 RSH server；
- ~user/.rhosts：规范出那个使用者可以不需要输入密码即可执行 RSH；
- /etc/pam.d/rsh：规范 root 能否使用 RSH 的设定档。

虽然 RSH 目前已经很少被使用，但是在内部主机的联机上面还是有他的存在的价值啦！ 因此，底下我们就来谈一谈如何玩弄这个 RSH 吧！

RSH Server

其实 R Shell 有很多的工具与启动的 port ，常见的 R Shell 工具有 rexec, rlogin, rsh 等， 而这些工具都对应到不同的 port 上面，你可以到 /etc/services 上查阅一下 512, 513, 514 这三个 port 吧！

- RSH Server 的启动：

如图 23 所示，我们在 RSH Client 计算机上面想要使用 RSH Server 上头的数据时， 那么 RSH server 自然就得要启动 RSH 这项服务啰！那么如何启动呢？简单的很～ 就利用 super deamon 来启动， 我们需要有 rsh 及 rsh-server 两个套件才行！请自行安装吧！ 另外， rsh-server 共提供三个服务，分别是『 rexec, rlogin, rsh 』，我们先单纯讨论 rsh 吧！ 安装完毕之后，直接启动即可：

```
[root@linux ~]# vi /etc/xinetd.d/rsh
service shell
{
        disable                 = no
        socket_type             = stream
        wait                    = no
        user                    = root
```

```
        log_on_success          += USERID
        log_on_failure          += USERID
        server                  = /usr/sbin/in.rshd
}
# 没错！只要将 disable 改成 no  即可！

[root@linux ~]# /etc/init.d/xinetd restart

[root@linux ~]# netstat -tlnp | grep 514
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State  PID/Program name
tcp        0      0 0.0.0.0:514    0.0.0.0:*       LISTEN 23369/xinetd
# 有看到 514 的 port 出现就对了！
```

- 设定可以使用 RSH 的来源主机与账号： /etc/hosts, /etc/hosts.equiv, ~user/.rhosts

假设我已经在 /etc/hosts 里面做好了我内部主机的 IP 与主机名称的对应了，例如，我 192.168.1.2 机器的 /etc/hosts 里面是这样的：

```
[root@linux ~]# vi /etc/hosts
127.0.0.1       localhost       localhost.localdomain
192.168.1.2     rsh.server      rshserver
192.168.1.100   rsh.client      rshclient
# 上面仅有两部主机，假设 RSH server 的 IP 是 192.168.1.2 啦！
```

这个档案很重要，因为我们的 RSH 通常是利用主机名称来作为指令的下达的， 所以啰，您局域网络内的主机名称与 IP 必须要设定的能够对应的起来，否则问题就大了。 在上表当中，你会发现到，其实鸟哥的重点是那部 rsh.server 的机器， 当我想要让 192.168.1.100 亦即是 rsh.client 那部机器连进来 rsh.server 执行一些指令的话， 那我就得要启动权限才行！此时就得要 /etc/hosts.equiv 来处理了。 这个档案的格式是这样的：

```
[root@linux ~]# vi /etc/hosts.equiv
rsh.client dmtsai
# 这个档案的格式是 [hostname] [username]
# 将你要开放的使用者与某主机给他对应好写上去即可！
```

这样就做好了设定了！未来任何想要登入这部 rsh.server 机器的主机，只要将他的主机名称与 IP 对应写入 /etc/hosts ，然后再将该主机名称写入 /etc/hosts.equiv ，就成功了！ 这个档案的设计只要是在规范『可以不用输入密码就能够进入本机执行指令』的设定啦！ 但是请注意，『在预设的情况下， root 是不允许使用 rsh 登入 rsh.server 机器的。』这个很重要！不要使用 root 一直测试啊！没有用的。 更多的 hosts.equiv 设定值，请参考 man hosts.equiv 啰！

但你可能会发现一件事，那就是每部主机与使用者都需要配合的话，但 /etc/hosts.equiv 预设仅有 root 可修改，如此一来实在不好管理！此时我们可以使用使用者家目录的档案来处理喔！那就是 ~/.rhosts 啰！这个档案的设定就更简单了！只要将使用者预计要登入的那部主机名称写入即可！

```
[root@linux ~]# vi ~dmtsai/.rhosts
```

```
rsh.client
```

这样就 OK 啦！意思是说，我这部 rsh.server 上有个 dmtsai 的使用者，他可以让 rsh.client 这个主机登入并且不需要密码即可进行 R Shell 的相关指令下达工作！ 同样的，如果我还想让其它使用者可以由不同的主机登入这部 rsh.server 进行 RSH 的话， 同样在他的家目录新增 .rhosts 这个档案即可！如果想要让未来新增的使用者都具有这个功能， 那就这样做：

```
[root@linux ~]# vi /etc/skel/.rhosts
rsh.client
```

未来新增使用者时，他们的家目录底下就会自动产生 .rhosts 的档案啰！ ^_^

- 让 root 也可以使用 RSH ：

先通知一声，鸟哥不建议您这样做！但是，如果万一您需要某些服务是 root 也需要的， 那或许就得要开放 root 使用 RSH 登入主机了。其实 root 不能使用 rsh 是因为 PAM 的问题而已，所以，你只要将 /etc/pam.d/rsh 这个档案批注掉一行即可：

```
[root@linux ~]# vi /etc/pam.d/rsh
#%PAM-1.0
# For root login to succeed here with pam_securetty, "rsh" must be
# listed in /etc/securetty.
auth       required     pam_nologin.so
#auth       required    pam_securetty.so
auth       required     pam_env.so
auth       required     pam_rhosts_auth.so
account    required     pam_stack.so service=system-auth
session    required     pam_stack.so service=system-auth
```

将上面这一行批注掉，立刻就能让 root 登入 rsh 主机啦！当然啦，如果您担心有问题， 那么修改另一个档案也可以：

```
[root@linux ~]# vi /etc/securetty
.....(省略).....
rsh
```

加入这一行也行！注意，上面这两个档案是具有相关性的，所以，您只要修改任何一个即可， 不必两个同时进行！然后将主机名称与 root 的对应写入 /etc/hosts.equiv 档案中即可！你的 root 就能够使用 rsh 啰！但是，鸟哥还是不很建议开放 root 使用 RSH 喔！注意注意！

RSH Client

这个 RSH client 的指令 rsh 预设是不给 root 执行的！所以不要使用 root 来测试！ 另外， RSH Client 最好与 RSH server 具有相同的账号来执行 rsh 比较不会有问题喔！

- 测试 RSH ：

接下来，当然要进行测试啰～请到 rsh.client 那部机器上面，使用 rsh 这个指令来下达指令吧！

```
[dmtsai@rshclient ~]$ rsh [-l 远程账号] [远程主机名] [远程主机指令]
参数：
-l    ：一般来说， server 与 client 『要有相同的使用者账号名称』比较好的！
        如果没有的话，那么您必须要指定 server 的使用者账号名才行！
远程主机名  ：您要登入的那部 rsh.server 主机名称，记得与 /etc/hosts 相应！
远程主机指令：您要在远程机器上面下达什么指令？

范例一：在 rsh.server 上面下达 ls -l / 这个指令：
[dmtsai@rshclient ~]$ rsh rsh.server 'ls -l /'
.....输出省略.....
# 注意喔，我是使用 dmtsai 这个一般身份使用者，而且 rshserver rshclient
# 两部主机上面都有一个名为 dmtsai 的使用者账号才行喔！至于那个 ls -l /
# 则是在 rsh.server 主机上面的指令！留意留意！
```

一般来说，由于 RSH server/client 最好是要有相同的账号，如此一来才能够避免很多不必要的权限问题 (permission denied.)。所以说，通常 RSH 可能会搭配后续会继续谈到的 NIS/NFS 等服务器才是！ 这样瞭呼？

另外， rsh 后续的指令通常仅适合单一指令而已，所以如果你的指令串很长 (接了很多参数)， 那最好将那一整串指令用单引号括起来，可以避免指令下达错误的问题喔！ ^_^

- 利用 rcp 复制：

除了 rsh 可以在远程直接操控系统外，我们可以透过 rcp 来进行复制喔！其实这个 rcp 与 scp 几乎一模一样啦！ 而且 rcp 的参数几乎与 cp 一模一样哩！另外， rcp 也是透过 RSH 这个 514 的 port 来进行数据的传输的。 简单的说明如下：

```
范例：先查阅远程主机有什么数据，然后将他复制过来：
[dmtsai@rshclient ~]# rsh rsh.server 'ls -l ~'
drwx------  3 dmtsai dmtsai     4096 Dec 27  2005 Desktop
-rw-r--r--  1 dmtsai dmtsai     3385 May 29 17:52 bashrc
drwx------  3 dmtsai dmtsai     4096 Mar  6  2006 mail
-rw-r--r--  1 dmtsai dmtsai   883888 May 29 17:51 netcdf.tar.gz
drwxr-xr-x  2 dmtsai dmtsai     4096 Jul 26 16:05 test
-rw-rw-r--  1 dmtsai dmtsai    34816 Mar 19  2006 testing.ppt

[dmtsai@rshclient ~]# rcp -r dmtsai@rsh.server:~/mail .
# 加上 -r 是为了要复制目录喔！否则的话，可以直接复制即可！
```

再说一次，其实这个 RSH 目前仅有在某些特殊的场合才会用到了！ 例如未来流行的新信息『 Cluster 』！不过如果要玩 Cluster 的话， 得要加入 NIS/NFS 等服务器哩！那就等您好好发展啦！ ^_^

以 rsync 进行同步镜相备份

我们曾在基础篇里面谈过 Linux 的备份策略， 该篇曾介绍常用的备份指令，包括 tar, dd, cp 等等，不过当时并未介绍网络，所以有个很棒的网络工具没有介绍， 那就是这个地方要谈到的 rsync 啦！这个

rsync 可以作为一个相当棒的异地备援系统的备份指令喔！ 因为 rsync 可以达到类似『镜相 (mirror) 』的功能呢！

rsync 最早是想要取代 rcp 这个指令的，因为 rsync 不但传输的速度快，而且他在传输时， 可以比对本地端与远程主机欲复制的档案内容，而仅复制两端有差异的档案而已，所以传输的时间就相对的降低很多！此外， rsync 的传输方式至少可以透过三种方式来运作：

- 在本机上直接运作，用法就与 cp 几乎一模一样，例如：
  rsync -av /etc /tmp (将 /etc/ 的资料备份到 /tmp/etc 内)

- 透过 rsh 或 ssh 的信道在 server / client 之间进行数据传输，例如：
  rsync -av -e ssh user@rsh.server:/etc /tmp (将 rsh.server 的 /etc 备份到本地主机的 /tmp 内)

- 直接透过 rsync 提供的服务 (daemon) 来传输，此时 rsync 主机需要启动 873 port：
  1. 你必须要在 server 端启动 rsync ， 看 /etc/xinetd.d/rsync 即可；
  2. 你必须编辑 /etc/rsyncd.conf 设定档；
  3. 你必须设定好 client 端联机的密码数据；
  4. 在 client 端可以利用：rsync -av user@hostname::/dir/path /local/path

其实三种传输模式差异在于有没有冒号 (:) 而已，本地端传输不需要冒号，透过 ssh 或 rsh 时，就得要利用一个冒号 (:)， 如果是透过 rsync daemon 的话，就得要两个冒号 (::) ，应该不难理解啦！因为本地端处理很简单， 而我们的系统本来就有提供 ssh 的服务，所以，底下鸟哥将直接介绍利用 rsync 透过 ssh 来备份的动作喔。 不过，在此之前咱们先来看看 rsync 的语法吧！

```
[root@linux ~]# rsync [-avrlptgoD] [-e ssh] [user@host:/dir] [/local/path]
参数：
-v ：观察模式，可以列出更多的信息；
-q ：与 -v  相反，安静模式，输出的信息比较少；
-r ：递归复制！可以针对『目录』来处理！很重要！
-u ：仅更新 (update)，不会覆盖目标的新档案；
-l ：复制连结文件的属性，而非连结的目标源文件内容；
-p ：复制时，连同属性 (permission) 也保存不变！
-g ：保存源文件的拥有群组；
-o ：保存源文件的拥有人；
-D ：保存源文件的装置属性 (device)
-t ：保存源文件的时间参数；
-I ：忽略更新时间 (mtime) 的属性，档案比对上会比较快速；
-z ：加上压缩的参数！
-e ：使用的信道协议，例如使用 ssh 通道，则 -e ssh
-a ：相当于 -rlptgoD ，所以这个 -a 是最常用的参数了！
更多说明请参考 man rsync 的解说！

范例一：将 /etc 的资料备份到 /tmp 底下：
[root@linux ~]# rsync -av /etc /tmp
```

```
....前面输出省略....
sent 23007335 bytes  received 32280 bytes  5119914.44 bytes/sec
total size is 22870014  speedup is 0.99
# 第一次运作时会花比较久的时间，因为首次建立嘛！如果再次备份呢？

[root@linux ~]# rsync -av /etc /tmp
building file list ... done
sent 77105 bytes  received 20 bytes  154250.00 bytes/sec
total size is 22870014  speedup is 296.53
# 瞧！立刻就跑完了！传输的数据也很少！因为再次比对，仅有差异的档案会被复制。

范例二：利用 dmtsai 的身份，将 rsh.server 使用者家目录复制到 /tmp
[root@linux ~]# rsync -av -e ssh dmtsai@rsh.server:~ /tmp
The authenticity of host 'rsh.server (192.168.1.2)' can't be established.
RSA key fingerprint is 29:b8:a9:32:ea:d8:ff:97:6c:42:3b:aa:11:ab:55:dd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'rsh.server' (RSA) to the list of known hosts.
dmtsai@rsh.server's password:
receiving file list ... done
....档案输出省略....
sent 8436 bytes  received 43224862 bytes  2789245.03 bytes/sec
total size is 43189031  speedup is 1.00

[root@linux ~]# ll -d /tmp/dmtsai
drwxr-xr-x  22 dmtsai dmtsai  4096 Sep 18 23:25 /tmp/dmtsai
# 瞧！这样就做好备份啦！很简单吧！
```

你可以利用上面的范例二来做为备份 script 的参考！不过要注意的是， 因为 rsync 是透过 ssh 来传输资料的，所以你可以针对 dmtsai 这个家伙制作出免用密码登入的 ssh 金钥！ 如此一来往后异地备援系统就能够自动的以 crontab 来进行备份了！简单到爆！

免密码的 ssh 账号我们在上头已经讲过了，撰写 shell script 的能力也是必须要有的！ 利用 rsync 来进行你的备份工作吧！ ^_^！ 至于更多的 rsync 用法可以参考本章后面所列出的参考数据网站喔！

重点回顾：

- 远程联机服务器可以让使用者在任何一部计算机登入主机，以使用主机的资源或管理与维护主机；
- 常见的远程登入服务有 rsh, telnet, ssh, vnc, 及 xdmcp 等；
- telnet 与 rsh 都是以明码传输数据，当数据在 Internet 上面传输时较不安全；
- telnet 与 rsh 预设无法让 root 的身份登入，不过可以藉由 pam 模块的修改而启用 root 登入功能；
- ssh 由于使用金钥系统，因此数据在 Internet 上面传输时是加密过的，所以较为安全；

- 但 ssh 还是属于比较危险的服务，请不要对整个 Internet 开放 ssh 的可登入权限，可利用 iptables 规范可登入范围；
- ssh 的 public Key 是放在主机端，而 private key 是放在 client 端；
- ssh 的联机机制有两种版本，建议使用可确认联机正确性的 version 2 ；
- 使用 ssh 时，尽量使用类似 email 的方式来登入，亦即： ssh username@hostname
- client 端可以比对 server 传来的 public key 的一致性，利用的档案为 ~user/.ssh/known_hosts；
- ssh 的 client 端软件提供 ssh, scp, sftp 等程序；
- 在 /etc/ssh/sshd_config 当中可以取消 root 的登入权限与修改支持的 ssh 金钥版本；
- 制作不需要密码的 ssh 账号可利用 ssh-keygen -t rsa 来制作 public, private Key pair；
- 上述指令所制作出的 public key 必须要上传到 server 的 ~user/.ssh/authorized_keys 档案中；
- 如果想以 X 图形系统登入 Linux 主机，则你必须要在 Client 主机启动 X server ， 需要在 Linux 主机启动 X client ；
- Xdmcp 是透过 X display manager (xdm, gdm, kdm 等) 所提供的功能协议；
- 若 client 端为 Linux 时，需要在 X 环境下以 xhost 增加可连接到本机 X Server 的 IP 才行；
- 除了 Xdmcp 之外，我们可以利用 VNC 来进行 X 的远程登入架构；
- VNC 预设开的 port number 为 5900 开始，每个 port 仅允许一个联机；
- 控制 rsh client 是否可以联机进入的设定档在 /etc/hosts.equiv 或 ~username/.rhosts ；
- rsh 支持的 client 端软件有 rsh, rlogin, rcp 等；
- rsync 可透过 ssh 的服务通道或 rsync --daemon 的方式来联机传输，其主要功能可以透过类似镜像备份， 仅备份新的数据，因此传输备份速度相当快速！

课后练习

- Telnet 与 SSH 都是远程联机服务器，为何我们都会推荐使用 SSH 而避免使用 Telnet 呢？原因何在？

  因为 Telnet 除了使用『明码』传送数据外，本身 telnet 就是很容易被入侵的一个服务器，所以当然也就比较危险了。 至于 ssh 其实也不是很安全的！由台湾计算机危机处理小组的文件可以明显的发现 openssl + openssh 也是常常有漏洞在发布！不过，比起 telnet 来说，确实是稍微安全一些！

- 请尝试说明 SSH 在 Server 与 Client 端联机时的封包加密机制；

  利用 key pair 来达到加密的机制：Server 提供 Public Key 给 Client 端演算 Private key ，以提供封包传送时的加密、解密！

- 请问 SSH 的设定档是哪一个？如果我要修改让 root 无法使用 SSH 联机进入我的 SSH 主机，应该如何设定？又，如果要让 badbird 这个使用者无法登入 SSH 主机，该如何设定？

SSH 设定档档名为 sshd_config ，通常放置在 /etc/ssh/sshd_config 内；如果不想让 root 登入，可以修改 sshd_config 内的参数成为：『PermitRootLogin no 』，并重新启动 ssh 来设定！如果要让 badbird 使用者无法登入，同样在 sshd_config 里面设定为：『DenyUsers badbird』即可！

- 在 Linux 上，预设的 Telnet 与 SSH 服务器使用的埠口(port number)各为多少？

  telnet 与 ssh 的埠口分别是：23 与 22！请参考 /etc/services 喔！

- 如果发现我无法在 Client 端使用 ssh 程序登入我的 Linux 主机，但是 Linux 主机却一切正常，可能的原因为何？(防火墙、known_hosts...)

  无法登入的原因可能有很多，最好先查询一下 /var/log/messages 里面的错误讯息来判断，当然，还有其它可能的原因为：

  1. 被防火墙挡住了，请以 iptables -L -n 来察看，当然也要察看 /etc/hosts.deny；
  2. 可能由于主机重新开机过， public key 改变了，请修改您的 ~/ssh/known_hosts 里面的主机 IP ；
  3. 可能由于 /etc/ssh/sshd_config 里面的设定问题，导致您这个使用者无法使用；
  4. 在 /etc/passwd 里面，您的 user 不具有可以登入的 shell ；
  5. 其它因素(如账号密码过期等等)

- 既然 ssh 是比较安全的资料封包传送方式，那么我就可以在 Internet 上面开放我的 Linux 主机的 SSH 服务了吗？！请说明您选择的答案的原因！

  最好不要对 Internet 开放您的 SSH 服务，因为 SSH 的加密函式库使用的是 openssl ，一般 Linux distribution 使用的 SSH 则是 openssh ，这两个套件事实上仍有不少的漏洞被发布过，因此，最好不要对 Internet 开放，毕竟 SSH 对于主机的使用权限是很高的！

- 如果我想要将 server 的重要资料备份到 backserver，如果可以使用 rsync 透过 ssh 的通道，你可以请尝试这样做：
  1. 先在 server 上面以 root 建置 ssh 的 public 与 private key pair ；
  2. 将 public key 传送与设定到 backserver 上的某个使用者环境下；
  3. 在 backserver 上面制作好预计要存放备份数据的目录！重点在于『权限』的设定上！
  4. 在 server 上面设定好 rsync 的备份 script ～
  5. 利用 crontab 每隔一段时间自动异地备份。

  请依照上述的作法在您的两部主机上面进行测试与实作。(未提供解答)

参考资源

- OpenSSH 官方网站：http://www.openssh.com/
- OpenSSL 官方网站：http://www.openssl.org/
- putty 官方网站：http://www.chiark.greenend.org.uk/~sgtatham/putty/

- pputty 中文网站：http://www.csie.ntu.edu.tw/~piaip/prjs/pputty/
- man vncserver
- man Xvnc
- 使用 X 的 VNC Modulehttp://phorum.study-area.org/viewtopic.php?t=25713
- http://fedoranews.org/tchung/vnc/03.shtml
- http://www.faqs.org/docs/Linux-HOWTO/XDMCP-HOWTO.html
- man rsh
- man rlogin
- 酷学园：用 rsync 做备份：http://phorum.study-area.org/viewtopic.php?t=15553
- 卧龙小三的 rsync 介绍：http://linux.tnc.edu.tw/techdoc/rsync.htm
- ADJ 实验室的 rsync + SSH：http://www.adj.idv.tw/server/linux_rsync.php

---

简易 NFS 服务器设定

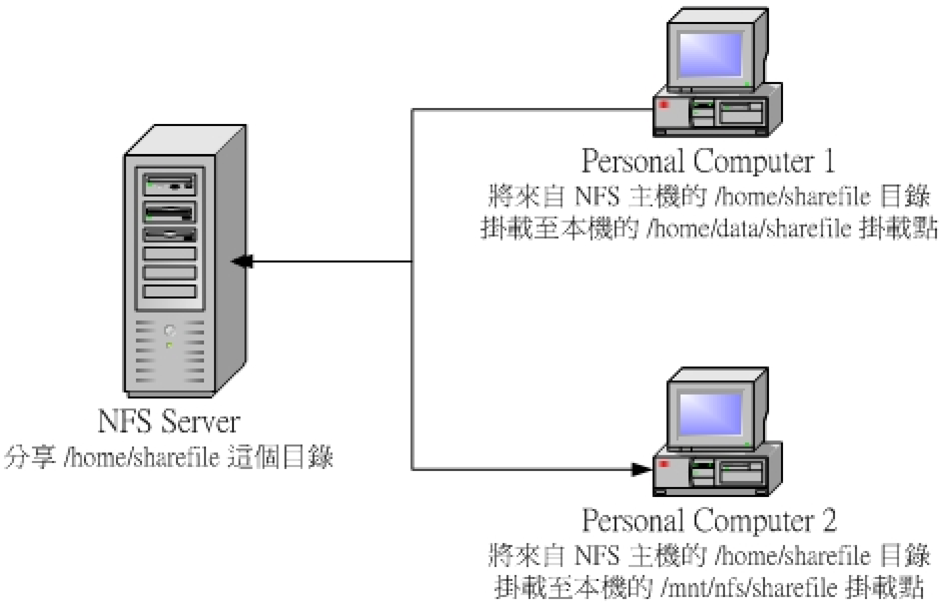
最近更新日期：2003/09/10

NFS 为 Network FileSystem 的简称，最早之前是由 Sun 这家公司所发展出来的，他的目的就是想让不同的机器、不同的操作系统可以彼此分享个别的档案啦！目前在 Unix Like 当中用来做为 file server 是相当不错的一个方案喔！基本上， Unix Like 主机连接到另一部 Unix Like 主机来分享彼此的档案时，使用 NFS 要比 SAMBA 快速且方便的多了！此外， NFS 的设定真的很简单，几乎只要记得启动 Remote Procedure Call 这个咚咚 (就是 portmap 这个套件啦！) 就一定可以架设的起来！真是不错啊！不过，如果要达成 Windows 与 Linux 之间的沟通，那么还是以 SAMBA 比较容易啊！无论如何， NFS 还是可以做为小公司或学校单位内部 Unix Like 机器共享 file 的一个 Server 喔！

NFS 的由来与其功能

什么是 NFS (Network FileSystem)

在开始进行 NFS 的设定之前，我们得先来了解一下，什么是 NFS 呢？不然讲了一堆也没有用，对吧！ ^_^！所谓的 NFS 就是 Network FileSystem 的缩写，最早之前是由 Sun 这家公司所发展出来的。他最大的功能就是可以透过网络，让不同的机器、不同的操作系统、可以彼此分享个别的档案 (share file)，所以，您也可以简单的将他看做是一个 file server 呢！这个 NFS Server 可以让您的 PC 来将网络远程的 NFS 主机分享的目录，挂载到本地端的机器当中，所以，在本地端的机器看起来，那个远程主机的目录就好像是自己的一个磁盘分割槽一样 (partition)！使用上面相当的便利！



图一、NFS 主机分享目录与 Client 挂载示意图

就如同上面的图示一般，当我们的 NFS Server 设定好了分享出来的 /home/sharefile 这个目录后，其它的 Client 端就可以将这个目录挂载到自己系统上面的某个挂载点(挂载点可以自订！)，例如前面图示中的 Personal Computer 1 与 Personal Computer 2 挂载的目录就不相同。我只要在 Personal Computer 1 系统中进入 /home/data/sharefile 内，就可以看到 NFS Server 系统内的 /home/sharefile 目录下的所有数据了 (当然，权限要足够啊！^_^)！这个 /home/data/sharefile 就好像我自己 Personal Computer 1 里面的一个 partition 喔！只要权限对了，那么您可以使用 cp, cd, mv, rm... 等等磁盘或档案相关的指令！真是他 X 的方便呐！

那么您或许会问啦：『咦！那么这个 NFS 是藉由什么样的协议来进行传输的呢？』虽然 NFS 有属于自己的协议与使用的 port number ，但是在数据传送或者其它相关讯息传递的时候， NFS 使用的则是一个称为远程过程调用(Remote Procedure Call, RPC)的协议来协助 NFS 本身的运作！

什么是 RPC (Remote Procedure Call)

那么什么是 RPC 呢？由字面上的意思来看『远程过程调用』不就是一些程序(Program)在执行远程联机时，需要用到的程序吗？呵呵！是这样没错啦！简单的来说，当我们在使用某些服务来进行远程联机的时候，有些信息，例如主机的 IP、服务的 port number、与对应到的服务之 PID 等等，都需要管理与对应！这些管理 port 的对应与服务相关性的工作，就是这个 Remote Procedure Call, RPC 的任务了！

好了，如果我们将 NFS 与 RPC 两者的相关性连接起来的话，那么您应该就可以知道： NFS 本身的服务并没有提供数据传递的协议，但是 NFS 却能让我们进行档案的分享，这其中的原因，就是 NFS 使用到一些其它相关的传输协议！而这些传输的协议，就是使用到这个所谓的 RPC 的功能啰！这也就是说， NFS 本身就是使用 RPC 的一个 program 就是了！说的更白话一点， NFS 也可以视作是一个 RPC server 啦！同时要注意到的是，在某些状况中，不但跑 NFS 的 Server 需要启动 RPC 的服务，连带的，要挂载 NFS partition 的 Client 机器，也需要同步启动 RPC 才行！

这样 Server 端与 Client 端才能藉由 RPC 的协议来进行 program port 的对应喔！

OK！简单的说， NFS 也可以看做是 RPC server 的一种，因为他是使用这种协议的 program 呀！^_^ ！那么为什么 NFS 要使用 RPC 执行呢？这是因为 NFS 本身可以被看做是一个档案系统，那么一来的话，您的使用者联机常常变化，而且您的档案内容啦、分享的目录啦，还有其它档案相关的信息等等，也都会常常在变化，这个时候，使用类似这种可以对应 program number 与 port number 的 RPC 就相当的方便了！也就是说，NFS 主要在管理分享出来的目录，而至于数据的传递，就直接将他丢给 RPC 的协议来运作就是了！

更多关于 NFS 协议的信息您可以参考底下的网页：

- http://www.faqs.org/rfcs/rfc1094.html
- http://www.tldp.org/HOWTO/NFS-HOWTO/index.html

NFS 启动的 RPC daemons

NFS server 总共需要启用到至少两个 daemons ，一个管理 Client 是否可以登入的问题，另一个管理登入主机后的 Client 能够使用的档案权限！如果您还要管理 quota 的话，那么 NFS 还会自动的再加载其它相关的 RPC program 呢！我们这里以最简单的方式来设定 NFS，说明如下：

- rpc.nfsd：这个 daemon 主要的功能就是在管理 Client 是否能够登入主机的权限啦，其中还包含这个登入者的 ID 的判别喔！

- rpc.mountd：这个 daemon 主要的功能，则是在管理 NFS 的档案系统哩！当 Client 端顺利的通过 rpc.nfsd 而登入主机之后，在他可以使用 NFS server 提供的档案之前，还会经过档案使用权限（ 就是那个 -rwxrwxrwx 与 owner, group 那几个权限啦 ） 的认证程序！他会去读 NFS 的设定档 /etc/exports 来比对 Client 的权限，当通过这一关之后， Client 就可以取得使用 NFS 档案的权限啦！(注：这个也是我们用来管理 NFS 分享之目录的使用权限与安全设定的地方哩！）

需要的套件

要启动 NFS 我们必须要有两个套件才行，分别是：

- nfs-utils 与 nfs-utils-clients (有时后仅有一个)
- portmap

- portmap：
  就如同刚刚提的到，我们的 NFS 其实可以被视为一个 RPC server program，而要启动任何一个 RPC server program 之前，我们都需要做好 port 的对应 （ mapping ） 的工作才行，这个工作其实就是『 portmap 』这个服务所负责的！也就是说，在启动任何一个 RPC server 之前，我们都需要启动 portmap 才行呢！那么这个 portmap 到底在干嘛呢？就如同这个服务的名称，哈哈！

就是作 port 的 mapping 啊！举个例子来说：当 Client 端尝试来使用 RPC server 所提供的服务时，由于 Client 需要取得一个可以连接的 port 才能够使用 RPC server 所提供的服务，因此， Client 首先就会去跟 portmap 讲『喂！可不可以通知一下，给我个 port number ，好让我可以跟 RPC 联络吧！』这个时候 portmap 就自动的将自己管理的 port mapping 告知 Client ，好让他可以连接上来 server 呢！所以啰：『启动 NFS 之前，请先启动 portmap ！』

- nfs-utils：
  就是提供 rpc.nfsd 及 rpc.mountd 这两个 NFS daemons 与其它相关 documents 与说明文件、执行档等的套件！这个就是 NFS 的主要套件啦！一定要有喔！

好了，知道我们需要这两个套件之后，现在干嘛？！赶快去您的系统先用 RPM 看一下有没有这两个套件啦！没有的话赶快用 RPM 去安装喔！不然就玩不下去了！

例题： 请问我的主机是以 RPM 为套件管理的 Linux distribution ，例如 Red Hat, Mandrake 与 OpenLinux 等版本，那么我要如何知道我的主机里面是否已经安装了 portmap 与 nfs 相关的套件呢？ 答： 简单的使用 rpm -qa ｜ grep nfs 与 rpm -qa ｜ grep portmap 即可知道啦！

Server 端的设定：

NFS 的套件结构

NFS 这个咚咚真的是很简单，上面我们提到的 NFS 套件中，设定档只有一个，执行文件也不多，记录文件也三三两两而已呐！赶紧先来看一看吧！ ^_^

- /etc/exports：这个档案就是 NFS 的主要设定档了！不过，系统并没有默认值，所以这个档案『不一定会存在』，所以您必须要使用 vi 主动的建立起这个档案喔！我们等一下要谈的设定也仅只是这个档案而已呐！

- /usr/sbin/exportfs：这个是维护 NFS 分享资源的指令，我们可以利用这个指令重新分享 /etc/exports 变更的目录资源、将 NFS Server 分享的目录卸载或重新分享等等，这个指令是 NFS 系统里面相当重要的一个喔！至于指令的用法我们在底下会再介绍。

- /usr/sbin/showmount：这是另一个重要的 NFS 指令。exportfs 是用在 NFS Server 端，而 showmount 则主要用在 Client 端。这个 showmount 可以用来察看 NFS 分享出来的目录资源喔！

- /var/lib/nfs/xtab：这个档案则是主要的 NFS 的纪录文件咯！当我们的 NFS 分享出目录资源后，到底有哪些 Client 端曾经连接上我们的 NFS 主机呢？呵呵！就是看这个档案的内容即可啰！ ^_^

就说不难吧！主要就是这几个啰！

主机的规划技巧建议

如果您的工作环境中，具有多部的 Linux 主机，并且预计彼此分享出目录时，那么在安装 Linux distribution 的时候，最好可以规划出一块 partition 作为预留之用。因为『 NFS 可以针对目录来分享』，因此，您可以将预留的 partition 挂载在任何一个挂载点，再将该挂载点(就是目录啦！)由 /etc/exports 的设定中分享出去，那么整个工作环境中的其它 Linux 主机就可以使用该 NFS 主机的那块预留的 partition 了！所以，在主机的规划上面，主要需要留意的只有 partition 而已。此外，由于分享的 partition 可能较容易被入侵，最好可以针对该 partition 设定比较严格的参数在 /etc/fstab 当中喔！

设定流程(/etc/exports)

我们在原理的部分对于 NFS 稍微解释了一下，哇！怎么看起来好像粉难喔！其实一点也不！为什么呢？因为 portmap 只要一支 scripts 就可以启动， NFS 只要设定一个档案就可以顺利运作！那么怎么能说不简单呢！呵呵！这个 NFS 真是他 X 的太太太.....简单了～在开始 NFS 之前，让我们先以 Windows 的系统当中的『资源共享』来说明一下整个流程吧：

5. 在 Windows Server 上面，开启档案总管，在某个目录上面按右键选择启动资源共享；
6. 在资源共享的内容当中，需要设定『使用者权限』(以 Windows 2000 为例)；
7. 在 Client 端需要登入 Windows server 时，需要启动『网络上的芳邻』来寻找可用的网络上面分享的目录，然后点选该目录，若可以登入该 Windows server 时，则可以依据步骤一的权限使用该目录下的档案！

呵呵！没错！ NFS 的整个流程也差不多是这样：

- 首先，需要确认一下您的 Linux 主机是否可以支持 NFS 这项服务，然后再设定一下使用者的来源 IP 或主机名称以及分享出去的目录的权限，之后呢，启动 NFS 即可将刚刚设定的目录给他分享出去了！
- 那么在 Client 端怎么使用这个分享出来的目录？就是先以 showmount 这支程序检查 Linux Server 是否有可以使用的 NFS 目录，如果有的话，就将他 mount 在本机上面，如果可以 mount ，那么就可以使用 NFS 主机提供的资源了！

哈哈！果然很简单吧！所以底下我们就来一个一个步骤的说明一下 NFS 怎么设定啰：

10. 系统需求

OK！每个咚咚的细部项目就来谈一谈吧：

- 系统需求：

  嘎！NFS 有最低硬件需求吗？呵呵！您误会了！这里的需求其实指的是『软件需求』啦！需要的是：

  1. 除了刚刚我们已经提到的两个套件『 portmap 与 nfs-utils 』必需要存在之外；
  2. 您的核心版本最好能够高于 2.2.xx 以后比较好！
  3. 此外，如果重新编译过核心，您必需『一定要选择』NFS 支持才行！

  目前，如果您使用的是安装时候的 Linux distribution 预设核心时，那么您都不用太担心，因为系统已经预设有支持 NFS 啰！所以底下的咚咚您都可以玩！但是，如果您已经重新编译过核心，并且不知道您是怎么编译的 (例如道听途说啦、试试看新鲜玩意啦等等的来编译您的核心时，所以没有注意到这个项目的选择) ，这个时候请拿出『鸟哥的 Linux 私房菜 -- 基础学习篇』好好的再次的读一遍『核心编译』！

- /etc/exports：

  好了，已经确认『一切 OK』之后，我们就真的要来玩弄 NFS 啦！这个东西真的很简单的啦，只要一个档案就可以搞定了！那就是编辑 /etc/exports 这个档案，请注意，这个档案如果不存在，请自行建立！并且，档名不要写错了喔！这个档案的内容很简单啦，我们列出他的规则：

```
[root@test root]# vi /etc/exports
[欲分享的目录]  [主机名称 1 或 IP1(参数 1,参数 2)] [主机名称 2 或 IP2(参数 3,参数 4)]
```

-

  上面的规则是这样的：[欲分享的目录]主要是要分享给[主机名称 1]及[主机名称 2]，但是提供给这两者的权限并不一样，其中，给主机名称 1 的权限是参数 1 与参数 2，至于给主机名称 2 的 Client 权限则是参数 3 与参数 4。好了，那么那个『权限』也就是『参数』主要有哪些呢？

  - rw：可擦写的权限；
  - ro：只读的权限；

- no_root_squash：登入 NFS 主机使用分享目录的使用者，如果是 root 的话，那么对于这个分享的目录来说，他就具有 root 的权限！这个项目『极不安全』，不建议使用！
- root_squash：在登入 NFS 主机使用分享之目录的使用者如果是 root 时，那么这个使用者的权限将被压缩成为匿名使用者，通常他的 UID 与 GID 都会变成 nobody 那个系统账号的身份；
- all_squash：不论登入 NFS 的使用者身份为何，他的身份都会被压缩成为匿名使用者，通常也就是 nobody 啦！
- anonuid：前面关于 *_squash 提到的匿名使用者的 UID 设定值，通常为 nobody，但是您可以自行设定这个 UID 的值！当然，这个 UID 必需要存在于您的 /etc/passwd 当中！
- anongid：同 anonuid ，但是变成 group ID 就是了！
- sync：数据同步写入到内存与硬盘当中；
- async：数据会先暂存于内存当中，而非直接写入硬盘！

大致的参数就是这几样啰！那么我们来假设几个例子好了：

- 思考一：我要将 /tmp 分享出去给大家使用，由于这个目录本来就是大家都可以读写的，因此我要让所有的人都可以存取。此外，我要让 root 写入的档案还是具有 root 的权限！那么您可以这么写喔！

```
[root@test root]# vi /etc/exports
/tmp  *(rw,no_root_squash)
```

- 这样一来，无论来自哪里(*万用字符！表示万事 OK！)都可以使用我的 /tmp 这个目录。请注意，那个 *(rw,no_root_squash) 中间没有空格符喔！而 /tmp 与 *(rw,no_root_squash) 则是有空格符来隔开的！特别注意到那个 no_root_squash 的功能！在这个例子中，如果您是 client 端，而且您是以 root 的身份登入您的 Linux 主机，那么当您 mount 上我这部主机的 /tmp 之后，您在该 mount 的目录当中，将具有『root 的权限！』

- 思考二：我要将一个公共的目录 /home/public 公开出去，但是只有限定我的局域网络内 192.168.0.0/24 这个网域可以读写，其它人则只能读取：

```
[root@test root]# vi /etc/exports
/tmp          *(rw,no_root_squash)
/home/public  192.168.0.*(rw)     *(ro)
/home/public  192.168.0.0/24(rw)  *(ro)
```

- 请注意，在上面的例子中，倒数两行的格式都可以适用！所以只要写一行即可！上面的例子说的是，当我的 IP 是在 192.168.0.0/24 这个网段的时候，那么当我在 Client 端挂载了 Server 端的 /home/public 后，针对这个被我挂载的目录我就具有可以读写的权限～至于如果我不是在这个网段之内，那么这个

目录的数据我就仅能读取而已，亦即为只读的属性啦！

- 思考三：我要将一个私人的目录 /home/test 开放给 192.168.0.100 这个 Client 端的机器来使用，那么我就必需这么写：

```
[root@test root]# vi /etc/exports
/tmp          *(rw,no_root_squash)
/home/public  192.168.0.*(rw)     *(ro)
/home/test    192.168.0.100(rw)
```

- 这样就设定完成了！而且，只有 192.168.0.100 这部机器才能对 /home/test 这个目录进行存取喔！

- 思考四：我要让 *.linux.org 网域的主机，登入我的 NFS 主机时，可以存取 /home/linux ，但是他们存数据的时候，我希望他们的 UID 与 GID 都变成 40 这个身份的使用者：

```
[root@test root]# vi /etc/exports
/tmp          *(rw,no_root_squash)
/home/public  192.168.0.*(rw)     *(ro)
/home/test    192.168.0.100(rw)
/home/linux   *.linux.org(rw,all_squash,anonuid=40,anongid=40)
```

- 特别注意到那个 all_squash 与 anonuid, anongid 的功能！如此一来，当 test.linux.org 登入这部 NFS 主机，并且在 /home/linux 写入档案时，该档案的所有人与所有群组，就会变成 /etc/passwd 里面对应的 UID 为 40 的那个身份的使用者了！

- 关于权限问题：
  无论任何时候，权限的问题都是需要考虑到的！让我们来看看刚刚建立的 /etc/exports 档案的内容：

```
[root@test root]# vi /etc/exports
/tmp          *(rw,no_root_squash)
/home/public  192.168.0.*(rw)     *(ro)
/home/test    192.168.0.100(rw)
/home/linux   *.linux.org(rw,all_squash,anonuid=40,anongid=40)
```

- 假设我在 192.168.0.100 登入这部 NFS (IP 假设为 192.168.0.2) 主机，并且我在 192.168.0.100 的账号为 test 这个身份，同时，在这部 NFS 上面也有 test 这个账号，果真如此的话，那么：
  1. 由于 192.168.0.2 这部 NFS 主机的 /tmp 权限为 -rwxrwxrwt ，所以我 (test 在 192.168.0.100 上面) 在 /tmp 底下具有存取的权限，并且写入的档案所有人为 test ；

2. 在 /home/public 当中，由于我有读写的权限，所以如果在 /home/public 这个目录的权限对于 test 有开放写入的话，那么我就可以读写，并且我写入的档案所有人是 test 。但是万一 /home/public 对于 test 这个使用者并没有开放可以写入的权限时，那么我还是没有办法写入档案喔！这点请特别留意！
3. 在 /home/test 当中，我的权限与 /home/public 相同的状态！还需要 NFS 主机的 /home/test 对于 test 有开放权限；
4. 在 /home/linux 当中就比较麻烦！因为不论您是何种 user ，您的身份一定会被变成 UID=40 这个账号！所以，这个目录就必需要针对 UID = 40 的那个账号名称，修改他的权限才行！

那么假如我在 192.168.0.100 的身份为 test2 ，但是 192.168.0.2 这部 NFS 主机却没有 test2 这个账号时，情况会变成怎样呢？

5. 我在 /tmp 底下还是可以写入，但是写入的档案所有人变成 nobody 了；
6. 我在 /home/public 里面是否可以写入，还需要视 /home/public 的权限而定，不过，反正我的身份就被变成 nobody 了就是；
7. /home/test 的观点与 /home/public 相同！
8. /home/linux 底下，我的身份就被变成 UID = 40 那个使用者就是了！

那么假如我在 192.168.0.100 的身份为 root 呢？ root 这个账号每个系统都会有呀！呵呵！权限变成怎样呢？

9. 我在 /tmp 里面可以写入，并且由于 no_root_squash 的参数，改变了预设的 root_squash 设定值，所以在 /tmp 写入的档案所有人为 root 喔！
10. 我在 /home/public 底下的身份还是被压缩成为 nobody 了！因为预设属性里面都具有 root_squash 呢！所以，如果 /home/public 有针对 nobody 开放写入权限时，那么我就可以写入，但是档案所有人变成 nobody 就是了！
11. /home/test 与 /home/public 相同；
12. /home/linux 的情况中，我 root 的身份也被压缩成为 UID = 40 的那个使用者了！

这样的权限讲解之后，您可以了解了吗？这里是最重要的地方，如果这一关通过了，底下的咚咚就没有问题啦！ ^_^

o 启动服务 portmap, nfsd

好了，设定 OK 也没有权限的问题之后（ 有问题也没关系，可以事后在好好的检视与修改一番！），再来自然就是启动他啰！如何启动呢？简单的很，直接给他 OK 下去！

```
[root@test root]# /etc/rc.d/init.d/portmap start<==启动 portmap !
[root@test root]# /etc/rc.d/init.d/nfs start     <==启动 NFS
```

- 那个 portmap 根本就不需要设定！只要直接启动他就可以啦！启动之后，会出现一个 port 111 的 sunrpc 的服务！那就是 portmap 啦！至于 nfs 则会启动至少两个以上的 daemon 出现！然后就开始在监听 Client 端的需求啦！启动之后，请赶快到 /var/log/messages 里面看看有没有被正确的启动呢？

```
[root@test root]# vi /var/log/messages
Nov 16 15:04:45 test portmap: portmap startup succeeded
Nov 16 15:04:53 test nfs: Starting NFS services:  succeeded
Nov 16 15:04:54 test nfs: rpc.rquotad startup succeeded
Nov 16 15:04:54 test nfs: rpc.mountd startup succeeded
Nov 16 15:04:54 test nfs: rpc.nfsd startup succeeded
```

- 要正常的出现上面的字样之后，才算是正确的启动喔！

- exportfs：
  好了，那么如果我们修改了 /etc/exports 这个档案之后，是否需要重新启动 nfs 呢？呵呵，并不需要，只要使用 exportfs 重新扫瞄一次 /etc/exports 这的档案，并且重新将设定加载即可！因此，就要来了解一下 exportfs 的用法了：

```
语法：
[root@test root]# exportfs [-aruv]
参数说明：
-a ： 全部挂载(或卸载) /etc/exports 档案内的设定
-r ： 重新挂载 /etc/exports 里面的设定，此外，亦同步更新 /etc/exports
     及 /var/lib/nfs/xtab 的内容！
-u ： 卸载某一目录
-v ： 在 export 的时候，将分享的目录显示到屏幕上！
范例：
[root@test root]# exportfs -rv  <==全部重新 export 一次！
exporting 192.168.0.100:/home/test
exporting 192.168.0.*:/home/public
exporting *.linux.org:/home/linux
exporting *:/home/public
exporting *:/tmp
reexporting 192.168.0.100:/home/test to kernel

[root@test root]# exportfs -au <==全部都卸载了！
```

- 要熟悉一下这个指令的用法喔！这样一来，就可以直接重新 export 我们的记录在 /etc/exports 的目录数据啰！

- 检验目录 /var/lib/nfs/xtab
  好了，当您顺利的将您的目录都分享出去之后，您怎么知道每个目录的分享权限呢？不

要忘记了，因为我们有相当多的预设属性呢！因此，这个时候就得需要检验一下您所分享的目录内容啰！看一下 /var/lib/nfs/xtab 这个档案吧！他有点像这样：

```
[root@test root]# vi /var/lib/nfs/xtab
/home/test  192.168.0.100(rw,sync,wdelay,hide,secure,root_squash,
no_all_squash,subtree_check,secure_locks,mapping=identity,anonuid=-2,
anongid=-2)
```

- 看到没？这个就是 /home/test 这个分享出去的目录的预设 NFS 里面的属性啦！这个属性状态里头有个比较奇怪的，那就是 anonuid=-2 这个，怎么有 uid=-2 的呢？呵呵！其实它说的是将 65536 - 2 的值，也就是 65534 的那个 UID 啦！对照一下 /etc/passwd ，您就会发现，哇！原来那就是 nobody 的啦！

- showmount：
  showmount 顾名思义，就是看看有没有可以 mount 的指令嘛！怎么用呢？

```
语法：
[root@test root]# showmount [-ae] hostname
-a ： 在屏幕上显示目前主机与 Client 所连上来的使用目录状态
-e ： 显示 hostname 这部机器的 /etc/exports 里面的分享目录！
范例：
[root@test root]# showmount -e localhost
Export list for localhost:
/tmp         *
/home/linux  *.linux.org
/home/public (everyone)
/home/test   192.168.0.100
```

- 很简单吧！所以，当您要扫瞄某一部主机他提供的 NFS 分享的目录时，就使用 showmount -e IP(或 hostname)即可！非常的方便吧！

- 观察启动的 port number：
  OK！来看看我们启动 NFS 之后，到底启动了多少的 port 呢？要注意的是，我们有启动 portmap 与 nfs 两支 scripts 喔！

```
[root@test root]# netstat -utln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN <==来自 portmap
tcp        0      0 0.0.0.0:817             0.0.0.0:*               LISTEN <==来自 rpc.xxxx
tcp        0      0 0.0.0.0:1266            0.0.0.0:*               LISTEN <==来自 rpc.xxxx
udp        0      0 0.0.0.0:2049            0.0.0.0:*                      <==就是 nfs 的 port
udp        0      0 0.0.0.0:814             0.0.0.0:*                      <==来自 rpc.xxxx
udp        0      0 0.0.0.0:1327            0.0.0.0:*                      <==来自 rpc.xxxx
```

```
udp        0      0 0.0.0.0:111              0.0.0.0:*                    <==来自 portmap
```

- 注意看到上面喔！总共产生了好多的 port 喔！真是可怕！先注意到 nfs 自己所开启的 port ，就是那个 2049 的 port 啦！那个就是 NFS 主要产生的 port 啰。那么其它的 rpc.xxxx 的 port 又是从何而来？ NFS server 在前面我们就提过了，他是 RPC server 的一种，而 NFS 由于提供了多个 program （ 例如 rpc.mountd, rpc.rquotad, rpc.nfsd... ），因此就需要启动多个 port 了！而且这些 port 是『随机产生的』，也就是那个 port number 不会是固定的啦！每次 restart nfs 都会得到不一样的 port number 呢！那么 Client 端怎么知道要连接上那个 port 来呼叫需要的 program 呢？呵呵！那就是 sunrpc (port 111) 那个 portmap 服务所产生的 port number 的功用啦！Client 会先连接到 sunrpc 那个 port 去知道应该到那个 port 去呼叫所需要的程序！所以啰， rpc.xxxx 等之类的 daemon 自然就不需要有固定的 port number 啰！

OK！这样一来， Server 端的设定就 OK 啦！

RPC server 的相关指令：

好了，既然我们知道这个 NFS 其实使用的是 RPC 这个咚咚，所以当然要知道 RPC 的每个 port 在干什么呀！这个时候，就不能不知道 rpcinfo 这个指令了！先来谈一谈这个指令的用法吧！

```
语法：
[root@test root]# rpcinfo [-p] hostname(orIP)
-p ：显示所有的 port 与 program 的信息！
范例：
[root@test root]# rpcinfo -p test.linux.org
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100011    1   udp   1014  rquotad
    100011    2   udp   1014  rquotad
    100011    1   tcp   1017  rquotad
    100011    2   tcp   1017  rquotad
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp   1339  nlockmgr
    100021    3   udp   1339  nlockmgr
    100021    4   udp   1339  nlockmgr
    100005    1   udp   1340  mountd
    100005    1   tcp   1271  mountd
    100005    2   udp   1340  mountd
    100005    2   tcp   1271  mountd
    100005    3   udp   1340  mountd
```

```
   100005    3   tcp   1271  mountd
```

这样就可以知道每个 port number 所对应的 program 啰！您也就知道这个 RPC server 提供给您的 program 是什么了！当然了，要让这个 rpcinfo 可以正确的动作，您的 portmap 得真的动起来才行呐！加油啰！

Client 端的设定：

挂载远程主机：

好了， Server 端已经设定完毕，接着下来自然就是要使用 Client 端连接上 Server 端啰！那么连接上 Server 的步骤是怎样呢？

1. 扫瞄可以使用的 Server 目录；
2. 在 Client 本地端建立 mount point；
3. 使用 mount 将远程主机分享的目录挂载进来；
4. 可能发生的问题解决(被防火墙挡掉了！？)。

OK 啦！所以我们得先知道一下我们的主机里面有什么？假设我的主机名称是 test.linux.org ，那么我要知道里头有些什么藉由 NFS 分享出来的目录，就给他 showmount 一下啰！

```
[root@test root]# showmount -e test.linux.org
Export list for localhost:
/tmp          *
/home/linux   *.linux.org
/home/public  (everyone)
/home/test    192.168.0.100
```

然后呢？假设我要将 /home/public 挂载在我的 /home/nfs/public 底下，那么我就得先有这个目录才行呀！然后再利用 mount 这个指令来挂载 /home/public 这个目录！有点像这样：

```
[root@test root]# mkdir -p /home/nfs/public <==建立 public 这个目录，加 -p 可以持续增加目录
[root@test root]# mount -t nfs test.linux.org:/home/public /home/nfs/public
挂载的格式：
[root@test root]# mount -t nfs hostname(orIP):/directory /mount/point
[root@test root]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1              1904920   1235380    572776  69% /
/dev/hdb1               976344    115212    810736  13% /backup
test.linux.org:/home/public
                       1904920   1235376    572776  69% /home/nfs/public <==这个是远程主机的容量
```

先注意一下挂载 NFS 档案的格式范例喔！呵呵！这样就可以将数据挂载进来啦！请注意喔！以后，只要您进入您的目录 /home/nfs/public 就等于到了 test.linux.org 那部远程主机的 /home/public 那个目录

中啰！很不错吧！那么如何将挂载的 NFS 目录卸载呢？就使用 umount 啊！

```
[root@test root]# umount /home/nfs/public
```

可能发生的问题：

通常无法挂载的原因有底下这几个：

1. 使用者的权限不符：以上面的例子来说明，我的 /home/test 只能提供 192.168.0.0/24 这个网域，所以，如果我在 test.linux.org 这部机器中，以 localhost 来挂载时，就会无法挂载上，这个权限概念没问题吧！那么您可以试试看：

```
[root @test root]# mount -t nfs localhost:/home/test /home/nfs
mount: localhost:/home/test failed, reason given by server: Permission denied
```

2. 所以啰！如果您发现上面的显示的讯息时，就表示您的主机权限不能够进入该目录啰！如果确定您的 IP 没有错误，那么请回到 /etc/exports 这个档案中，针对您自己的 IP 来进行修正吧！

3. 忘记启动 portmap ：
   这个最容易被忘记了!就是忘记了启动 portmap 这个服务啦!如果您发现您的 mount 的讯息是这样：

```
[root@test root]# mount -t nfs localhost:/home/test /home/nfs
mount: RPC: Port mapper failure - RPC: Unable to receive
```

4. 或者是：

```
[root@test root]# mount -t nfs localhost:/home/test /home/nfs
mount: RPC: Program not registered
```

5. 那么就赶紧将 portmap 启动吧！！并且也需要将 nfs 重新启动喔！

```
[root@test root]# /etc/rc.d/init.d/portmap start
[root@test root]# /etc/rc.d/init.d/nfs restart
```

6.
7. 被防火墙挡掉了：
   这个也很容易忘记了!那就是重新设定一下您的防火墙,这包含了两部份,包括 iptables 与 TCP_Wrappers ！因为我们启动了 portmap ，这个东西有两个数据需要分享出来，一个是 port 111 需要提供出去，因此您的 iptables 规则当中，需要开放这个 port 喔！有点像这样的几行字要加入您的 iptables rules 当中：

```
iptables -A INPUT -p TCP --dport 111 -j ACCEPT
iptables -A INPUT -p UDP --dport 111 -j ACCEPT
```

8. 如果您已经开放了这个 port 的连接权限，却还是无法连接成功，那么应该就是 TCP_Wrappers 的问题了！检查一下您的 /etc/hosts.deny 里头是否有这行：

```
[root@test root]# vi /etc/hosts.deny
ALL: ALL
```

9. 果真如此的话，由于 portmap 是由 portmap 这个 daemon 所启动的，所以您就必须要在 /etc/hosts.allow 里面加入这一行：

```
[root@test root]# vi /etc/hosts.allow
portmap: ALL
```

10. 或者是将 ALL 改成您所想要让他使用 NFS 的网域即可！这样说可以了解了吗？若想进一步了解一下防火墙，请参考前面章节提过的：简易防火墙建置。

关机或结束时的注意事项：

需要注意的是，由于 NFS 使用的这个 RPC 在 client 端连上主机时，那么您的主机想要关机，那可就会成为『不可能的任务』！如果您的 Server 上面还有 Client 在联机，那么您要关机，可能得要等到数个钟头才能够正常的关机成功！嘎！真的假的！不相信吗？不然您自个儿试试看！ ^_^！所以啰，建议您的 NFS Server 想要关机之前，能先『关掉 portmap 与 nfs 』这两个东西！如果无法正确的将这两个 daemons 关掉，那么先以 netstat -utlp 找出 PID ，然后以 kill 将他关掉先！这样才有办法正常的关机成功喔！这个请特别特别的注意呢！

安全设定(被防火墙挡掉了)：

好了！一些注意事项讲完了之后，再来呢？对了！又是最重要的安全设定方面的问题了！那么 NFS 可以设定安全的地方有哪里呢？其实还不少呢？由外而内可以这样看：

1. iptables 防火墙设定；
2. TCP_Wrappers 防火墙设定；
3. /etc/exports 权限设定。

防火墙的基本概念请参考『简易防火墙建置』一文，最好能将该篇文章给他看完，否则还真难了解底下在干嘛～嗯！假设您已经看完该篇短文了，接着下来我们就得要一步一步的接着建立防火墙啰！

- 使用 iptables 限制大范围联机：
  假设我们的 NFS 主要是针对内部网络开放而已，而对于外部网络只有对学术网络开放，亦即是 140.0.0.0/8 ，那么您可以使用这样的语法：

```
iptables -A INPUT -i eth0 -p TCP -s 192.168.0.0/24 --dport 111 -j ACCEPT
```

```
iptables -A INPUT -i eth0 -p UDP -s 192.168.0.0/24 --dport 111 -j ACCEPT
iptables -A INPUT -i eth0 -p TCP -s 140.0.0.0/8    --dport 111 -j ACCEPT
iptables -A INPUT -i eth0 -p UDP -s 140.0.0.0/8    --dport 111 -j ACCEPT
```

- 这样大致上就可以让 192.168.0.0/24 这个 C Class 的网域与 140.0.0.0/8 这个 A Class 的网域到您的主机里面来，而其它的联机就视您的原本的 iptables 的状态而定喔！

- 使用 TCP_Wrappers 限制更细的范围：
  事实上，如果您不懂得如何设定 iptables 的话，那也没关系，我们可以使用 TCP_Wrappers 阿！因为要使用 NFS 就必须要通过 portmap 这一关(因为要使用 RPC 啦！)，而这个 portmap 可以藉由 TCP_Wrappers 来管理！呵呵！太好了！那么就将他联机的范围限制的更小啰！我们可以在 /etc/hosts.allow 里面规定连上 NFS 主机的主机 IP 与名称，假设限制中的主机只有 192.168.0.0/24 这个 C class 及 140.116.44.125 这个主机，以及后面接的是 ncku.edu.tw 的网域可以连上我的 NFS 主机，那么我可以写成这样：

```
[root@test root]# vi /etc/hosts.allow
portmap: 192.168.0.0/255.255.255.0
portmap: 140.116.44.125
portmap: .ncku.edu.tw

[root@test root]# vi /etc/hosts.deny
portmap: ALL
```

- 呵呵！这样可就设定好啰！很简单的吧！

- 使用 /etc/exports 设定更安全的权限：
  这就牵涉到您的逻辑思考了！怎么设定都没有关系，但是在『便利』与『安全』之间，要找到您的平衡点呐！善用 root_squash 及 all_squash 等功能，再利用 anonuid 等等的设定来规范登入您主机的使用者身份！应该还是有办法提供一个较为安全的 NFS 主机的！

- Client 端挂载的问题：
  基本上，在 Client 端挂载的时候，为了担心会不小心刚 NFS 端挂进来的具有 SUID 权限档案的程序执行！这个很可能会危害到系统的安全呢！因为 SUID 本来就不是很安全的嘛！所以呢，您这个 root 也可以将 NFS 所分享的目录以较为安全的情况挂载进来！例如：

```
[root@test root]# mount -t nfs -o nosuid,ro hostname:/directory /mount/point
```

- 选择 nosuid 也是一个很不错的抉择喔！

通常我们都会约略的建议，不要启动 NFS Server ，即使要启动，最好也是针对某个范围来进行目录的分享！并且，『要分使用者层级来管理』会比较好一些喔！底下我们就来实际的在您的机器上面搞一个简单的 NFS server 吧！

实际演练：

假设环境：

1. 假设我的 Linux 主机为 192.168.0.100 这一部；
2. 预计将 /tmp 以可擦写，并且不限制使用者身份的方式分享给所有 192.168.0.0/24 这个网域中的所有 Linux 工作站；
3. 预计开放 /home/nfs 这个目录，使用的属性为只读，可提供除了网域内的工作站外，向外亦提供数据内容；
4. 预计开放 /home/upload 做为 192.168.0.0/24 这个网域的数据上传目录，其中，这个 /home/upload 的使用者及所属群组为 nfs-upload 这个名字，他的 UID 与 GID 均为 210；
5. 预计将 /home/andy 这个目录仅分享给 192.168.0.50 这部 Linux 主机，以提供该主机上面 andy 这个使用者来使用，也就是说， andy 在 192.168.0.50 及 192.168.0.100 均有账号，且账号均为 andy ，所以预计开放 /home/andy 给 andy 使用他的家目录啦！

实地演练：

好了，那么请您先不要看底下的答案，先自己动笔或者直接在自己的机器上面动手作作看，等到得到您要的答案之后，在看底下的说明吧！

- 首先，就是要建立 /etc/exports 这个档案的内容啰，您可以这样写吧！

```
[root @test root]# vi /etc/exports
/tmp          192.168.0.*(rw,no_root_squash)
/home/nfs     192.168.0.*(ro)  *(ro,all_squash)
/home/upload  192.168.0.*(rw,all_squash,anonuid=210,anongid=210)
/home/andy    192.168.0.50(rw)
```

- 大概就是这样子吧！您可以自行测试看看！

- 再来，就是要建立每个对应的目录的实际 Linux 权限了！我们一个一个来看：

```
1. /tmp
[root @test root]# ll /
drwxrwxrwt    6 root     root         4096 Nov 16 09:07 tmp

2. /home/nfs
[root @test root]# mkdir -p /home/nfs           <==建立所需要的目录
[root @test root]# chmod 755 -R /home/nfs      <==修改较为严格的档案权限
将目录与档案设定成只读！不能写入的状态，会更保险一点！

3. /home/upload
[root @test root]# groupadd -g 210 nfs-upload  <==先建立所需要的 210 这个群组
[root @test root]# useradd -g 210 -u 210 -M nfs-upload <==建立需要的使用者名称
[root @test root]# mkdir -p /home/upload       <==建立起目录了！
[root @test root]# chown -R nfs-upload:nfs-upload /home/upload <==修改拥有者！
```

```
如此，则使用者与目录的权限都设定妥当啰！

4. /home/andy
[root @test root]# ll /home
drwx------    3 andy     andy         4096 Oct 28 13:37 andy
```

- 这样子一来，权限的问题大概就可以解决啰！
- 启动 portmap 与 nfs 服务：

```
[root @test root]# /etc/rc.d/init.d/portmap start
[root @test root]# /etc/rc.d/init.d/nfs start
```

-
- 在 192.168.0.50 这部机器上面演练一下：

```
1. 确认可用目录
[andy @linux50 andy]$ showmount -e 192.168.0.100
Export list for 192.168.0.100:
/tmp         192.168.0.*
/home/nfs    (everyone)
/home/upload 192.168.0.*
/home/andy   192.168.0.50

2. 建立挂载点：
[andy @linux50 andy]$ mkdir -p /home/zzz/tmp
[andy @linux50 andy]$ mkdir -p /home/zzz/nfs
[andy @linux50 andy]$ mkdir -p /home/zzz/upload
[andy @linux50 andy]$ mkdir -p /home/zzz/andy

3. 实际挂载：
[andy @linux50 andy]$ su   <==通常 Linux 只允许 root 来挂载！
[root @linux50 andy]# mount -t nfs 192.168.0.100:/tmp /home/zzz/tmp
[root @linux50 andy]# mount -t nfs 192.168.0.100:/home/nfs /home/zzz/nfs
[root @linux50 andy]# mount -t nfs 192.168.0.100:/home/upload /home/zzz/upload
[root @linux50 andy]# mount -t nfs 192.168.0.100:/home/andy /home/zzz/andy
[root @linux50 andy]# exit
```

整个步骤大致上就是这样呐！加油喔！

重点回顾

- Network FileSystem (NFS) 可以让主机之间透过网络分享彼此的档案与目录；
- NFS 主要是透过 RPC 来进行 file share 的目的，所以 Server 与 Client 的 RPC 一定要启动才行！

- NFS 主机可以控制联机的 Client 端的登入与权限；
- NFS 的设定档就是 /etc/exports 这个档案；
- NFS 的重要登录档可以参考 /var/lib/nfs/xtab 这个档案，还包含相当多有用的信息在其中！
- NFS 主机要关机之前，请务必先关闭 portmap 与 nfs server ，否则关机无法顺利成功；
- NFS 主机在更动 /etc/exports 这个档案之后，可以透过 exportfs 这个指令来重新挂载分享的目录！
- 可以使用 rpcinfo 来观察 RPC program 之间的关系！！！
- NFS 主机在设定之初，就必须要考虑到 client 端登入的权限问题，很多时候无法写入或者无法进行分享，主要是 Linux 实体档案的权限设定问题所致！
- NFS 的防火墙设定可以透过控制 RPC 的主要 port ，亦即是 111 这个 port 来管理！此外，亦可透过 TCP_Wrappers 来管理！
- NFS 客户端只要成功 mount NFS 主机分享的目录之后，使用上面就好像自己的 partition 一般；
- NFS 客户端可以透过使用 showmount, mount 与 umount 来使用 NFS 主机提供的分享的目录！

本章与 LPI 的关系

在 LPI 网站 http://www.lpi.org 里面提到的，关于 NFS 的考试题库的地方，只有在 LPI level 1 的 102 ，里面的 topic 113 Networking Services ，第四点当中，简易的 NFS 设定。强调的是『应试者需了解 NFS 的设定、启动与关闭的关系』至于会考的档案与指令可能有这些：

- /etc/exports
- /etc/fstab
- mount
- umount

参考资源：

- http://www.faqs.org/rfcs/rfc1094.html
- http://www.tldp.org/HOWTO/NFS-HOWTO/index.html
- man exports

本章习题练习

- NFS 的主要设定档为何？而在该档案内主要设定项目为何？
- 在 NFS 主要的设定档当中仅有少许的参数说明，至于预设的参数说明则没有在该档案当中出现，请问，如果要查阅更详细的分享出来的档案的属性，要看那个档案？
- 如果已经启动了 nfs 这个服务器，但是却又修改过主要设定档，请问可以使用那个指令来重新挂载分享出来的目录与 client 端权限的设定值？
- 在 client 端如果要挂载 NFS 所提供分享的档案，可以使用那个指令？
- 在 NFS 主要设定档当中，可以透过那个参数来控制不让 client 端以 root 的身份使用您所分享出来的目录与档案？

我在 client 端挂载了 NFS Server 的某个目录在我的 /home/data 底下，当我执行其中某个程序时，却发现我的系统被破坏了？您认为可能的原因为何？该如何克服这样的问题，尤其是当我的 Client 端主机其实是多人共享的环境，怕其它的使用者也同样发生类似的问题呢？！

简易 DHCP 服务器设定

最近更新日期：2003/03/15

> 如果您在工作单位使用的是笔记型计算机，而且常常要带着您的笔记型计算机到处跑，那么由前几章的『连上 Internet 』设定当中，会发现，哇！我的网络卡参数要常常修改啊！而且，每到一个新的地方，就得问清楚该地的 Server 提供的网络参数才行！真是麻烦～～这个时候，动态主机设定协议 (DHCP) 可就大大的派上用场啦！DHCP 这个服务器可以自动的分配 IP 与相关的网络参数给 Client 端，来提供 Client 端自动以主机提供的参数来设定他们的网络，如此一来，使用者只要将自己的 Notebook 设定好经由 DHCP 协议来取得网络参数后，一插上网络线，呵呵！马上就可以享受 Internet 的服务啦！很方便吧！所以得来瞧一瞧这个好用的协定喔！

原理：

老规矩，在正式的进入 DHCP (Dynamic Host Configuration Protocol) 主机设定之前，我们先来认识一下 DHCP 这个协议吧！还有，需要了解的是，我们是否有需要『一定』得设定 DHCP 这个服务器呢？这里都需要厘清一下概念喔！

什么是 DHCP：

在开始 DHCP 的说明之前，我们先来复习一下之前在『网络基础』里面提到的几个网络参数吧！要设定好一个网络的环境，使计算机可以顺利的连上 Internet ，那么您的计算机里面一定要有底下几个网络的参数才行，分别是：

- IP
- netmask
- network
- broadcast
- gateway
- DNS IP

其中，那个 IP, netmask, network, broadcast 与 gateway 都可以在 /etc/sysconfig/network-scripts/ifcfg-eth[0-n] 这些个档案里面设定，DNS 的地址则是在 /etc/resolv.conf 里头设定。呵呵！只要这几个项目设定正确，那么计算机应该就没问题的可以上网了！所以说，您家里面的 3, 4 部计算机，您都可以手动的来设定好您所需要的网络参数，然后利用 NAT 主机的功能，就可以大摇大摆的 连上 Internet 了！真是不错 ^_^，不是吗？

好了，现在让我们换一个大一些些的场景吧！假设您是学校宿舍的网络管理员，所管理的学生计算机大概有 100 部好了，那么您怎么设定好这 100 部的计算机呢？

7. 直接每一部计算机都让您登门拜访手动的去设定好？
8. 将所有的学生都集合起来，然后精神训话.....喔不！是直接教导一下怎么设定？还是
9. 藉由一部主机来自动的分配所有的网络参数给宿舍内的任何一部计算机？

这三种解决方案所需要的时间都不相同，如果您选择的是(1)，那么我个人认为，您不是工作狂就是疯掉了，因为所要花费的时间与您所得的薪水与付出的心力是完全不成比例的～～如果选择是(2)那么很可能您会被挂上独裁者、没良心的管理员的称号！如果是选择(3)呢？恭喜您！这个方案的管理时间花费最短，也是最不麻烦的作法啦！

呵呵！知道我要说些什么了吗？是的！这个 DHCP (Dynamic Host Configuration Protocol) 主机最主要的工作，就是在进行前面提到的第三个方案，也就是自动的将网络参数正确的分配给网域中的每部计算机，让 Client 端的计算机可以在开机的时候就立即自动的设定好网络的参数值，这些参数值可以包括了 IP、netmask、network、gateway 与 DNS 的地址等等。如此一来，呵呵！身为管理员的您，只要注意到这一部提供网络参数的主机有没有挂掉就好了，其它同学们的个人计算机，哈！您想都不必想要怎么去帮忙！因为 DHCP 主机已经完全都帮您搞定啦！ ^_^！ 阿！当管理员最大的幸福就是可以喝喝茶、聊聊天就能控管好一切的网络问题呢！

DHCP 的运作方式：

运作模式：
那么 DHCP 是怎么运作的呢？现在假设我们的机器在同一个网域当中，也就是说， DHCP Server 与他的 Clients 都在同一个网段之内，可以透过软件广播的方式来达到相互沟通的状态。那么 Client 藉由 DHCP Server 得到 IP 的程序为：

10. 若 Client 端计算机设定使用 DHCP 协议以取得网络参数时，则 Client 端计算机在开机的时候，或者是重新启动网络卡的时候，会自动的发出 DHCP Client 的需求给网域内的每部计算机：这个时候，由于发出的讯息希望每部计算机都可以接受，所以该讯息除了网络卡的硬件地址(MAC)无法改变外，需要将该讯息的来源软件地址设定为 0.0.0.0 ，而目的地址则为 255.255.255.255 (这个我们 Linux 会自动帮您设定，无须考虑这个问题！)。这个时候，网域内的其它没有提供 DHCP 服务的计算机，收到这个封包之后会自

动的将该封包丢弃而不回应；而如果是 DHCP 主机呢？

11. DHCP 主机响应讯息：如果是 DHCP 主机收到这个 Client 的 DHCP 需求时，那么 DHCP 主机首先会针对该次需求的讯息所携带的 MAC 与 DHCP 主机本身的设定值去比对，如果 DHCP 主机的设定有针对该 MAC 做静态 IP (每次都给予一个固定的 IP)的提供时，则提供 Client 端相关的固定 IP 与相关的网络参数；而如果该讯息的 MAC 并不在 DHCP 主机的设定之内时，则 DHCP 主机会选取目前网域内没有使用的 IP(这个 IP 与设定值有关) 来发放给 client 端使用!此外,需要特别留意的是,在 DHCP 主机发放给 Client 端的讯息当中，会附带一个『租约期限』的讯息，以告诉 Client 端，您这个 IP 可以使用的期限有多长！

12. Client 端接受来自 DHCP 主机的网络参数,并设定 Client 自己的网络环境:当 Client 端接受响应的讯息之后，首先会以 ARP 封包在网域内发出讯息，以确定来自 DHCP 主机发放的 IP 并没有被占用!如果该 IP 已经被占用了,那么 Client 对于这次的 DHCP 信息将不接受，而将再次向网域内发出 DHCP 的需求广播封包；若该 IP 没有被占用，则 client 可以接受 DHCP 主机所给的网络的参数,那么这些参数将会被使用于 client 端的网络设定当中，同时， Client 端也会对 DHCP 主机发出确认封包，告诉 Server 这次的需求已经确认！而 Server 也会将该信息记录下来；

13. Client 端结束该 IP 的使用权：当 Client 开始使用这个 DHCP 发放的 IP 之后，有几个情况下他可能会失去这个 IP 的使用权：
    - Client 端离线：不论是关闭网络接口(ifdown)、重新开机 (reboot)、关机 (shutdown) 等行为，皆算是离线状态，这个时候 Server 端就会将该 IP 回收，并放到 Server 自己的备用区中，等待未来的使用；
    - Client 端租约到期：前面提到 DHCP server 端发放的 IP 有使用的期限，Client 使用这个 IP 到达期限规定的时间，就需要将 IP 缴回去！这个时候就会造成断线，而 Client 也可以再向 DHCP 主机要求再次分配 IP 啰！

以上就是 DHCP 这个协议在 Server 端与 Client 端的运作状态，由上面这个运作状态来看，我们可以晓得，喝！只要 Server 端设定没有问题，加上 Server 与 Client 在硬件联机上面确定是 OK 的，那么 Client 就可以直接藉由 Server 来取得上网的网络参数，当然啦，只要我们这些管理员能够好好的、正确的管理好我们的 DHCP ，嘿嘿！那么自然上网的设定就变成一件很简单的事情啦！

IP 取得的方法：
在上面的步骤里面，注意到第二步骤了吗？就是 DHCP 会去比较 MAC 这个硬件地址，并判断该 MAC 是否需要给予一个固定的 IP 呢！呵呵！所以啦，我们在 Client 端由 DHCP 主机取得的 IP 主要有两种方式：

- 静态 (Static) IP ：只要那个 client 端计算机的网络卡不换掉，那么 MAC 肯定就不会改变，由于 DHCP 可以根据 MAC 来给予固定的 IP ，所以该计算机每次都能以一个固定的 IP 连上 Internet ！呵呵！这种情况比较适合当这部计算机需要用来做为提供

区域内的一些网络服务的主机之用。那么如何在 Linux 上面知道您的 MAC 呢？很简单啦！有很多的方式，最简单的方式就是使用 ifconfig 及 arp 来进行：

```
[root@test root]# ifconfig eth0
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:FC:22:9C:57
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:333678 errors:0 dropped:0 overruns:0 frame:0
          TX packets:377219 errors:0 dropped:0 overruns:0 carrier:0
          collisions:195 txqueuelen:100
          RX bytes:42243563 (40.2 Mb)  TX bytes:278373316 (265.4 Mb)
          Interrupt:10 Base address:0x6100

[root@test root]# arp
Address             HWtype  HWaddress           Flags Mask            Iface
test.linux.org      ether   00:50:FC:22:9C:57   C                     eth0
192.168.1.100       ether   00:02:44:19:A6:AD   C                     eth0
```

-
- 动态 (dynamic) IP：Client 端每次连上 DHCP 所取得的 IP 都不是固定的！都直接经由 DHCP 所随机由尚未被使用的 IP 中提供！

除非您的局域网络内的计算机有可能用来做为主机之用，所以必需要设定成为固定 IP ，否则使用动态 IP 的设定比较简单，而且使用上面具有较佳的弹性。怎么说呢？假如您是一个 ISP 好了，而您只申请到 150 个 IP 来做为您的客户联机之用。那么您是否真的只能邀集到 150 的使用者？呵呵！当然不啰！我可以邀集 200 个使用者以上呢！为什么？这样想好了，我今天开了一家餐馆，里面只有 20 个座位，那么是否我一餐只能卖给 20 个人呢？当然不是啦！因为客人是人来人往的，有人先吃有人后吃，所以同样是 20 个座位，但是可以有 40 个人来吃我的简餐，因为来的时间不一样嘛！了解了吗？呵呵！对啦！您这个 ISP 虽然只有 150 个 IP 可以发放，但是因为您的使用者并非 24 小时都挂在在线的，所以您可以将这 150 个 IP 做良好的分配，让 200 个人来『轮流使用』这 150 个 IP 哩！

好了，那么另外一个问题，还是搞不懂什么是『静态 IP 』与『动态 IP』呢？不都是由 DHCP 发放的吗？让我们再来谈一个小例子好了。目前(2003 年)上网的主流是 ADSL 拨接制这种方法，其中，由于拨接制所以每次上线的 IP 都不一样！这可以想成是 DHCP 的 动态 IP 分配方式！那么如果您是使用 GIGA 的拨接制给予的固定 IP 呢？呵呵！那个方式其实还是使用拨接之后才能得到联机啦！只不过 ISP 经由 PPP 协议当中的密码来判断使用者，让同一个使用者每次都可以收到一个固定的 IP 而已!这样可以理解了吗？呵呵!您可以想成,拨接到 ISP 就是类似 client 端发送一个 DHCP 的需求给 DHCP 主机，而将 GIGA 想成我们 DHCP 里面的固定 IP 分配方式，而 Seednet 之类的 ISP 动态给予 IP 的，就是 DHCP 里面使用的 dynamic IP 分配方式啦！（ 注：其实在软件地址亦即是 IP 上面，只有 Public IP 与 Private IP 两种，中文翻译成『公共 IP』与『私有 IP』这两个，至于其它所谓的『静态 IP』、『实体 IP』、『虚拟 IP』、『浮动式 IP』

等等，都是藉由一些 IP 取得的方式来分类的，对于此种分类方式可能会造成读者的困扰，所以，请特别留意这些 IP 的意义，不要搞混了！如果还是害怕会搞不清楚，那么只要先记得『公共 IP』是可以直接与 Internet 相互沟通的，至于『私有 IP』则是不能直接与 Internet 沟通的内部 IP 段！）

关于租约的行为：
怪了！如果我们观察上面 DHCP 运作模式的第二个步骤，您会发现最后面还有一个租约期限！干嘛还要这样的一个期限呢？其实设定期限还是有个优点啦！最大的优点就是可以避免该 IP 被某些使用者一直占用着，但该使用者却是 Idle（ 发呆 ）的状态！举个例子来说，我们刚刚不是说到，我有 150 个 IP ，但是偏偏我有 200 个用户嘛！那么假设刚好例如 2002 年的世界杯足球赛好了，每个使用者都急着上网知道消息，那么将会达到交通尖峰时段！也就是说，这 200 个人同时要来使用这 150 个 IP ，有可能吗？当然不可能！肯定会有 50 个人无法联机，因为『很抱歉！目前系统正在忙线中，请您稍后再拨！』那怎么办？这个时候租约到期的方式就很有用处啦！那几个已经联机进来很久的人，就会因为租约到期而被迫离线，这个时候该 IP 就会被释放出来，哈哈！大家赶快抢呀！先抢到先赢喔！所以，那 50 个人（ 包括被迫离线的那个朋友 ）只好继续的、努力的、加油的来进行 DHCP 的要求啰！ ^_^""

虽然说是优点，但是其实如果站在使用者的角度来看，还是可能会造成公愤的！凭什么大家一起交钱，我先联机进来就需要先被踢出去？～呵呵！这个在早期 Hinet 就是这点被骂的要死！为什么呢？因为他的 ADSL 拨接制，似乎真的就有这个租约到期的问题，限制的时间似乎是 24 小时的样子！所以，使用 Hinet ADSL 拨接制的朋友，每 24 小时就要忍受一次断线！我没有使用过 Hinet 的 ADSL 拨接制，网络上的朋友确实有响应这样的事情，但是我不确定现在（ 2003/03/15)Hinet 是否还是用这套设定值？这样您可以了解租约到期的行为了吗？！ ^_^

关于 DHCP 主机个数：
或许您曾经发现过一件事情，那就是，当我的网域里面有两部以上的 DHCP 主机时，到底哪一部主机会设定我的这部 Client 端计算机？呵呵！很抱歉，俺也不晓得！因为在网络上面，很多时候都是『先抢先赢』的，同样的， DHCP 的回应也是如此！当 Server1 先响应时，您使用的就是 Server1 所提供的网络参数内容，如果是 Server2 先响应，您就是使用 Server2 的参数来设定您的 PC ！不过，前提之下当然是这些计算机的『物理联机』都是在一起的啊！

什么时候需要 DHCP ？

什么时候才需要架设 DHCP 呢？是否每个人都最好架设一部 DHCP 主机呢？那可就见仁见智啦！接下来要告知大家的是几个概念性的问题，您倒不一定『必需』遵守底下的一些概念呢！反正，自己的网域自己『爽』就好啦！

- 什么时候最好使用 DHCP ？
  在某些情况之下，倒是强烈的建议架设 DHCP 主机的！什么情况呢？例如：

  1. 您的公司内部很多 Notebook 计算机使用的场合！因为这种 NoteBook 本身就是移动性的装置，如果每到一个地方都要去问人家『喂！您这边的网络参数是

什么？』还得要担心是否会跟人家的 IP 相冲突等等的问题!这个时候,DHCP 可就是您的救星啰！因为 Notebook 在使用上，当设定为 DHCP client 的时候，那么只要他连接上的网域里面有一部可以动作的 DHCP ，那么那部 notebook 就可以连接上 Internet 了！真好，不是吗？！

2. 网域内计算机数量相当的多时：另外一个情况就是您所负责的网域内计算机数量相当庞大时，大到您没有办法一个一个的进行说明来设定他们自己的网络参数，这个时候为了省麻烦，还是架设 DHCP 来的方便呐！况且，维护一部您熟悉的 DHCP 主机，要比造访几十个不懂计算机的人要简单的多哩！^_^

- 什么情况下不建议使用 DHCP 主机？
  虽然 DHCP 有很多好处，但是您有没有发现一个步骤怪怪的呀！回头看一下那个步骤一，Client 在开机的时候会主动的发送讯息给网域上的所有机器，这个时候，如果网域上就是没有 DHCP 主机呢？很抱歉，那么您的这部 Client 端计算机，『仍然会持续的发送讯息！』真正的时间与次数我不晓得会有多久，不过，肯定会超过 30 秒以上，甚至可以达到一分钟以上！哇！那么这段时间您能干嘛？呵呵！除了等、还是等！所以啰，如果计算机数不多，还是使用手动的方式来设定一下就好了！方便嘛！

0. 在您网域内的计算机，有很多机器其实是做为主机的用途，很少 Client 需求，那么似乎就没有必要架设 DHCP ；

1. 更极端的情况是，像一般家里，只有 3~4 部计算机，这个时候，架设 DHCP 只能拿来练练功力，事实上，并没有多大的效益；

2. 当您管理的网域当中，大多网络卡都属于老旧的型号，并不支持 DHCP 的协议时；

3. 很多使用者的信息知识都很高，那么也没有需要架设 DHCP 啦。

如前所述，上面的都是概念性的说法，事实上，一件事情的解决之道是有很多的方案的，没有所谓的『完全正确』的方案，只有『相对可行、并且符合经济效益与功能』的方案！所以啰，架设任何网站之前，请先多评估评估呐！

套件安装：
在 Linux 上面 DHCP 套件的安装也是很简单的，不需要以 Tarball 来安装啦！直接拿出您的原版光盘，mount 他，并且找到 dhcp 字样的套件，使用 RPM 安装好就好了！以 mandrake 9.0 及 Red Hat 9 为例的话，您需要的 DHCP 套件为：

```
# 在 Mandrake 9.0 当中：
[root@test root]# rpm -qa | grep dhcp
dhcp-server-3.0-1rc9.2mdk
```

```
dhcp-common-3.0-1rc9.2mdk
dhcp-client-3.0-1rc9.2mdk

# 在 Red Hat 9 当中：
[root@test root]# rpm -qa | grep dhcp
dhcp-3.0pl1-23
dhcp-devel-3.0pl1-23
```

看到了吧!我们需要的套件在不同的 Linux 版本上面会有些许的差异喔!如果是 Mandrake 系列的话,DHCP 套件档案总数会有三个啰！如果没有安装的话，请拿出您的原板光盘(不论是那个 Linux distribution ，应该都是这三个套件名称啦！只是后面的版本可能不一样就是了！但是，需要注意的是，在 Red Hat 上面，这三个套件被整合成为一个，亦即是 dhcp-xxxx.rpm 的档案就是了！)，先挂载上去，然后就以 rpm 给他安装啦！什么？不知道如何搞定 RPM ？唉啊！又得拿出『鸟哥的 Linux 私房菜 -- 基础学习篇』好好的 K 一 K 『 RPM 与 Tarball 的使用』。另外，如果在系统当中没有找到 dhcp 套件的话，还可以使用前面 网络升级套件 章节提到的 apt-get 或者是 urpmi 都是很好的安装方式呐！ ^_^

设定 DHCP Server：

DHCP 套件结构

在 DHCP 的套件结构当中，也是仅有一个设定档。这个设定档不见得会存在喔！也是需要我们手动来建立的呐！谈一谈先：

- /etc/dhcpd.conf：这个就是 dhcp 的主要设定档咯！这个档案不见得会存在，请手动来建立喔！另外，其实每个 dhcp 套件在释出的时候，都会附上一个范例档案，您可以使用 rpm -ql grep dhcp 来查询到 dhcpd.conf.sample 这个档案呐！然后将该档案复制成为 /etc/dhcpd.conf 后，在手动去修改 /etc/dhcpd.conf 即可，这样设定比较容易咯！(注：这个设定档在不同的 Linux distribution 当中会有不一样的放置目录喔！例如在 openlinux 底下，这个设定档预设需要放置在 /etc/dhcpd/dhcpd.conf 呢！)
- /usr/sbin/dhcpd：这个就是 DHCP 的 daemon 执行档啰！
- /var/lib/dhcp/dhcpd.leases：这档案颇有趣的！我们前面原理部分不是有提到『租约』吗？呵呵！租约的启始与到期日就是记录在这个档案当中的咯！

整个 DHCP 的设定档与检查的档案就是这几个而已，很简单吧！^_^

主机的规划技巧

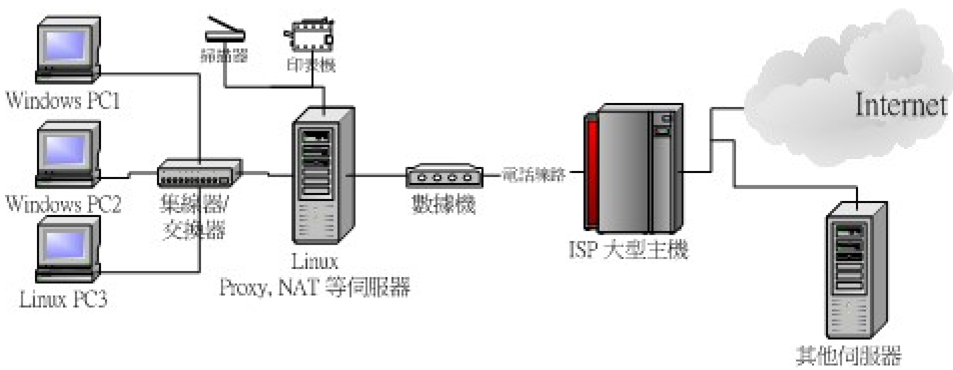
如果您的机器仅要提供 DHCP 的服务的话，那么真的一部 P-133 MMX 的 586 主机就很足够了！而且 partition 的规划不是很重要，因为 DHCP 主机并不会使用到硬盘空间，最多仅只是那个租

约到期的纪录文件而以(/var/lib/dhcp/dhcpd.leases)。

设定流程

好不容易！终于到了要架设 DHCP Server 的时候了，哇！好感动.....咦！鸟哥怎么老是在唱单口相声.... @_@。其实要设定好一个 DHCP 主机还真是蛮简单的，只要设定一个档案即可！但是前面的确认工作请先仔细的查验好！

4. 确认硬件的联机没有问题：
   这可是架设网站的第一个重点！如果您的硬件没有搞定，那么软件再怎么厉害，呵呵！也是没有用的啦！目前我的架构就像底下这样，我的 Linux 主机上面有两张网络卡，一张对内一张对外喔！



5. 确认 dhcp 相关设定档案摆放的路径：
   请您特别留意的是，不同的 Linux distribution 中，每个套件的设定文件放置的位置都不相同，例如 OpenLinux 使用的 dhcp 设定档放置在 /etc/dhcpd/dhcpd.conf ，而 Red Hat 与 Mandrake 则放在 /etc/dhcpd.conf 里头！那么我要怎么确认呢？呵呵，因为我知道设定档案与 server 有关，所以当然是查询 dhcp-server 这个套件啰，要查询设定档，可以加上 c 的参数，所以：

```
[root@test root]# rpm -qc dhcp-server (在 Red Hat 使用 rpm -qc dhcp )
/etc/dhcpd.conf.sample
/etc/rc.d/init.d/dhcpd
/etc/sysconfig/dhcpd
/var/lib/dhcp/dhcpd.leases
```

6. 喝！知道那个 /etc/dhcpd.conf.sample 是范例文件，注意，我们的 DHCP 设定档之档名为 dhcpd.conf ，所以啰，马上就知道我们的设定档案即为 /etc/dhcpd.conf 啰！咦！这个档案不存在哪！没有关系，我们要自动的建立这个档案喔！除此之外，如果您是使用 RPM 安装的 DHCP ，那么察看一下 /etc/rc.d/init.d/dhcpd 这个 scripts 的内容，

也可以知道设定参数文件的位置，甚至也可以修正 dhcpd.conf 的位置呢！

7. 设定 dhcpd.conf 设定档：
   好了，那么来到最重点啦！我们的 DHCP 就只要设定这个档案即可！那么这个档案如何设定呢？基本上，我们刚刚前面提过说， DHCP 的 IP 分配可分为给予动态 IP 与静态 IP ，其中，又需要了解的是，如果需要设定静态 IP 的话，那么就必须要知道要设定成静态 IP 的那部计算机的硬件地址 (MAC)才行，请使用 arp 及 ifconfig 来查知您的接口的 MAC 吧！此外，我们需要设定的项目大概有几项：

   - 整体设定(Global)：里面含有租约期限啦、或许还有 DNS 地址与 router 的设定等等内容；
   - 动态 IP 设定：使用 subnet 的项目与 range 的参数来设定要分配出去的 IP ！请先确认好您的网段喔！
   - 静态 IP 设定：使用 host 这个项目段，配合 MAC 来设定！

   在 dhcpd.conf 这个档案里头有些地方要特别留意：

   - 『#』为批注符号；
   - 除了括号那一行之外，其它的每一行后面都要以『 ; 』做为结尾！这很重要！

   那么我的网络环境是怎样呢？鸟哥假设是这样的喔：

   - 目前我的内部网段设定为 192.168.1.0/24 这一段，且 router 为 192.168.1.2 ，此外， DNS 主机的 IP 为中华电信的 168.95.1.1 及 Seednet 的 139.175.10.20 这两个；
   - 我想要让每个使用者预设租约为 3 天，最长为 6 天；
   - 我只想要分配的 IP 只有 192.168.1.21 到 192.168.1.100 这几个，其它的 IP 则保留下来；
   - 我还有一部主机，他的 MAC 是『 00:40:95:30:43:B4 』，我要给他的主机名称为 vbird-inside ，且 IP 为 192.168.1.5 这个。

   则，我的设定档为：

```
[root@test root]# vi /etc/dhcpd.conf
# 这个档案在各家 Linux 中，可能放置的目录不同而且不见得会存在，请自行建立！

# 再次说明，这个档案中，行首为 # 的，则为批注，且，
# 设定的每一行之行尾，都需要 ; 符号。
```

```
# 这个档案的设定写法主要有两种，首先是有独特的设定名称的，例如
# default-lease-time ，
# < 参数代号 >        < 设定内容 > ;
# default-lease-time   259200       ;
# 再来则是由 option 这个代号后面接的参数代码
# option < 参数代码 > < 设定内容 >  ;
# option domain-naim  "domain.name" ;

# 1. Global 首先建立整体的设定值，亦即是，当 subnet 或 host
#    当中没有设定的，以 global 为准

# 1.1 设定租约期限：
default-lease-time 259200;  # 预设的租约时间，后面接的时间为秒钟。
max-lease-time     518400;   # 但是超过租约时间(就是 default-lease-time)
                             # 还可续约，不过，最长还是只到 518400 秒！

# 1.2 设定领域名称与 DNS IP
option domain-name   "dhcp.vbird.org" ; # 设定每部主机的后头
                                        # 领域名称为 dhcp.vbird.org ！
option domain-name-servers 168.95.1.1, 139.175.10.20 ;
# 上面这个设定在写入 DNS 的 IP ，会自动在 Client 端修改 /etc/resolv.conf
# 档案，若有两个 DNS 的 IP 以上，那么就需要以 , 符号分隔开来！
# 最后面还是得加上 ; 符号
ddns-update-style ad-hoc;
# 在 Mandrake 9.0 这一版当中，由于加入许多新功能，所以，这一行
# 就需要加入才行启动 DHCP ，至于在 Red Hat 当中则是加入如下行：
# ddns-update-style interim;

# 2. 动态 IP 分配的设定：
subnet 192.168.1.0 netmask 255.255.255.0 {
      range 192.168.1.21 192.168.1.100;
      option broadcast-address 192.168.1.255;
      option routers 192.168.1.2;
}
# 动态 IP 就如同上面的设定内容，主要以 subnet 与 netmask 来进行前头的说明，
# 亦即先宣告我要的网段是哪一段，这部份就一定得要回去参考一下网络基础
# 的部分了！千万记得呐！然后，那个 range IP1 IP2; 那一行，都以空格符
# 隔开即可，而 range 表示我要分配的 IP 就是在 IP1 与 IP2 这之间的 IP 段
# 就是了！再者，我总是要给 broadcast 吧！没错！那个
# option broadcast-address 即是给予的 broadcast 了；至于 routers
# 不用说也知道那是什么了吧？所以，我们就已经含有 IP, network, netmask,
# broadcast 与 router 啰！这样就可以顺利上网啦！至于 DNS 则是在
# 上面 global 处已经设定啰！
```

```
# 3. 静态 IP 的设定：
host vbird-inside {
      hardware ethernet  00:40:95:30:43:B4;
      fixed-address 192.168.1.5;
      option broadcast-address 192.168.1.255;
      option routers 192.168.1.2;
}
# 静态 IP 肯定跟 MAC 有关的，所以请先查出来 MAC 吧！然后的设定您都知道啦！
```

- 在 Global 的设定当中，也就是不在 subnet 与 host 的括号内的设定数据，就是可以被视为『预设』的数据啦！也就是说，当 subnet 里头没有写到租约期限啦、没有写到 domain-name 啦，那么这个 Global 的设定就会自动被拿去做为该设定内的设定内容啰！

8. 建立租约期限档案：
   既然我们知道 DHCP 是由用户与主机端之间的租约是否到期来进行是否继续联机的动作，那么自然就有所谓的『签约仪式』啰！哈哈！真会掰！这个时候，我们就得要知道一下，那么我到底要在哪里设定这个租约期限档案呢？如果在比较早期的版本中，这个步骤一定要进行的！不过，目前的版本中，似乎预设已经有这个租约期限档案了！无论如何，还是作一下比较安心啦！

```
[root@test root]# touch /etc/dhcpd.leases
# 建立租约档案同样的，注意您的版本目录！
```

9.
   这个档案倒是蛮有趣的！因为在实际的运作过程中，这个档案本身不会有什么作用，但是在启动了 DHCP 之后，这个档案会被 copy 一份成为底下的档案：『/var/lib/dhcp/hdcpd.leases』而真正在记录的，其实就是这个 /var 底下的档案啦！后面我们再来看一下这个档案的内容吧！目前还不会用到这个档案的内容啰！(注：如果您使用的 Linux 是最近的版本，那么应该不需要再执行这个步骤了！无论如何，您可以先略过这个步骤，到启动的时刻下，如果未能正确启动，再回来做这一步都还来得及呢！)

10. 编辑 scripts 内容：
   咦！不是可以启动了吗？喔不～我们还得要检查一下，您要提供 DHCP 的接口是哪一个呢！就如同我上面的图示，基本上，我的主机是有两块网络卡的，一块对外一块对内，而我只对这个对内部的网络卡启动 DHCP ，因此，我可以修改一下我的 /etc/rc.d/init.d/dhcp 这个档案：注：我是以 Mandrake 9.0 为例来说明的，如果您是使用 Red Hat 或者是 OpenLinux 等其它版本时，您应该可以找到『 daemon

/usr/sbin/dhcpd 』那一行，将他改成底下的样子也就可以啦！

```
[root@test root]# vi /etc/rc.d/init.d/dhcpd
# 这是启动的 script 档案

# 先找到底下的设定内容，如果没有找到也没有关系，就自行新增吧！
CONFIGFILE="/etc/dhcpd.conf"           # 这个是设定档案的完整路径名称
LEASEFILE="/var/lib/dhcp/dhcpd.leases"  # 这个是在设定租约期限记录档案
INTERFACES="eth0"                    # 这个则是要启用 DHCP 的主机网络适配卡
OPTIONS="-q"                           # 其它的 dhcpd 的参数设定值！
start() {
        .....(略)...
        daemon /usr/sbin/dhcpd -cf $CONFIGFILE -lf $LEASEFILE $OPTIONS $INTERFACES
        .....(略)...
}
....
```

11.

基本上，上面的 /etc/dhcpd.conf 设定档案的所在目录是可以变动的！就在这里进行更动即可！您可以使用 man dhcpd 就可以知道每个参数代表的意义了！请注意，那个 start() 里面的信息，只要更动找到的 daemon.... 那一行即可！至于上面的参数设定内容，则可以参考 BASH 的变量设定方法来写！目前因为我们只要针对 eth0 来启用 DHCP 而已，所以上面 INTERFACES 的部分我就写入了 eth0 而已，如果您还要启动 eth1 的话，那么就需要：『 INTERFACES="eth0 eth1"』两个界面中间以空格符来隔开！这样，大致上就已经设定妥当了！准备来去启动啦！

12. 启动 DHCP 服务：
启动服务的方法不需要再教了吧！？直接给他执行刚刚改过的 scripts 即可：

```
[root@test root]# /etc/rc.d/init.d/dhcpd start 启动的啦！
Starting dhcpd:                                    [  OK  ]
```

13.

注意屏幕前面要显示 OK 才可以呢！好了，除此之外，我们还要确认 DHCP 已经启动才行！

14. 确认正确的启动 DHCP ：观察启动的 port 号与登录文件的讯息：
要注意的是，虽然我们已经驱动了 script 了，不过，您仍然不会知道，那个 script 是否真的让您的 service 启动了哪？所以，在启动任何的服务之后，观察一下两个东西，一个是 port 是否以启动，另一个则是到登录文件去查询一下讯息！这都是很重要的信息呢！所以，您应该这样做：

```
1. 最重要的就是察看登录档! 也就是 /var/log/messages 这个档案了!
[root@test root]# vi /var/log/messages  先看看登录挡吧!
Nov 23 23:35:09 vbird dhcpd: Wrote 0 deleted host decls to leases file.
Nov 23 23:35:09 vbird dhcpd: Wrote 0 new dynamic host decls to leases file.
Nov 23 23:35:09 vbird dhcpd: Wrote 0 leases to leases file.
Nov 23 23:35:09 vbird dhcpd: dhcpd startup succeeded
一定要看到 Wrote xxxx to leases file 以及 succeeded 的字眼,
才能确定启动成功!

2. 再来, 观察一下 port 有没有在 listen
[root@test root]# netstat -utl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 *:bootps                *:*
[root@test dhcp]# netstat -utln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:67              0.0.0.0:*
```

15.

仔细的看到喔! DHCP 显示的是 bootps 这个字样! WHY ? 没有什么奇特的原因啦，因为 DHCP 的前身就是 bootps 这个 protocol ，所以当然就沿用啦! 如果您想要修改这个字眼使成为 dhcp 的话，可以修改 /etc/services ：

```
[root@test root]# vi /etc/services
找到这两行:
bootps          67/tcp                          # BOOTP server
bootps          67/udp

将他改成
dhcp            67/tcp
dhcp            67/udp
```

16.

那未来使用 netstat -tln 就可以得到 dhcp 的显示了! 嗄! 忘记 netstat 后面的参数意义! 喂! 不要再混了! 赶快回到前几篇提过的『Linux 常用网络指令介绍』练一练基本功吧!

17. 修改 /etc/hosts 档案内容对应:

如果您有仔细的瞧过前几章的 网络基础 的话，那么应该还会记得那个 /etc/hosts 会影响内部计算机的联机速度很大吧? ! 那么我现在使用 DHCP 之后，糟糕! 我怎么知道哪一部 PC 连上我的主机，那么要怎么填写 /etc/hosts 的内容呢? 这真是太简单了! 那就将所有可能的计算机 IP 都加进去该档案呀! ^_^ ! 以我为例，在这个例子中，我

的分配的 IP 至少有 192.168.1.5, 192.168.1.21 ~ 192.168.1.100 ，所以我的 /etc/hosts 可以写成：

```
[root@test root]# vi /etc/hosts
127.0.0.1      localhost.localdomain localhost
192.168.1.2    vbird-server
192.168.1.5    static-ip
192.168.1.21   dynamic-021
192.168.1.22   dynamic-022
.....
192.168.1.100 dynamic-100
```

18.
这样一来，所有可能连进来的 IP 都已经有纪录了，哈哈！当然没有什么大问题啰！ ^_^

设定 DHCP Client ：
DHCP 的 Client 端，可以是 Windows 也可以是 Linux 呢！由于我的领域内刚好有两部 Client 端的计算机，一部为 Linux (Red Hat 7.3) 另一部为 Windows 2000 ，这里就提一下，分别是怎样设定的呢？

- Linux 的 DHCP Client 端设定：
  设定还真是简单的不行～直接修改一个档案即可！由于我的 Linux 计算机有两块网络卡，其中，eth0 为使用 DHCP 来启动的，所以我就可以修改底下的档案呢：

```
[root@test root]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=dhcp    <==只要这一行设定对了，其它的不要管！
BROADCAST=192.168.1.255
IPADDR=192.168.1.235
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
GATEWAY=192.168.1.2
```

- 需要特别强调的是，Mandrake 与 Red Hat 都使用『BOOTPROTO=dhcp』来设定的！但是， OpenLinux 却是使用『DYNAMIC=dhcp』来设定的！所以要注意您的 Linux distribution 喔！改完之后，就将我们的网络卡关掉再开！请注意，如果您是在远程进行这个动作，您的联机『肯定会挂掉！』，因为网络卡被您关了嘛！呵呵！所以请在本机前面才进行喔！

```
[root@test root]# ifdown eth0; ifup eth0
```

```
Determining IP information for eth0 ..... done
[root@test root]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:54:DG:08:QE:BE
          inet addr:192.168.1.100  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:590 errors:0 dropped:0 overruns:0 frame:0
          TX packets:208 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:55145 (53.8 Kb)  TX bytes:29113 (28.4 Kb)
          Interrupt:10 Base address:0xd000
```

-

  棒吧！已经正确的取得 IP 的资料啰！

- Windows 2000 底下的 DHCP Client 设定方式：

  在 Windows 底下的设定也真是太简单了！

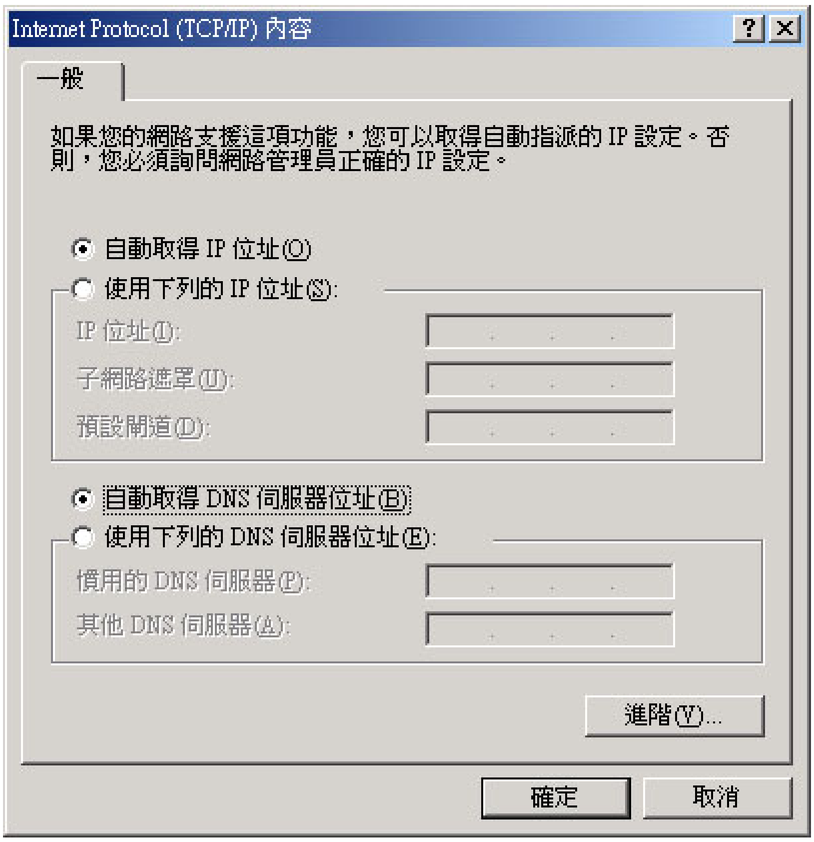
  1. 直接在『网络上的芳邻』按右键，选内容；
  2. 然后选择『区域联机』那一项，之后会出现如下的图示：



  3. 在上图当中，按下『内容』则会出现下面的图示：



4. 然后选择『Internet Protocol (TCP/IP) 』那一项，之后按下内容，会出现选项：



5. 然后自然就是一直按下『确定』！直到回到正常的桌面为止！这样就已经正确的启动了！哇！就是这么简单！
6. 手动修订一下网络设定的方式：
   - 在 Windows 底下，要修订 IP 的方式依据不同的版本而有不同！如果是 Windows 98 系列的版本，就需要使用『winipcfg』，出现的是窗口画面，您可以自己调一下；
   - 在 Windows 2000 底下，我不知道怎么叫出窗口画面，所以直接开启一个『 C:> 提示字符』，在『开始』==>『程序集』==>『附属应用程序』里面的『命令提示字符』那个就是啦！

```
C:\>ipconfig /all   <==秀出所有的属性

Windows 2000 IP Configuration

Ethernet adapter 区域联机:

        Connection-specific DNS Suffix  . : dhcp.vbird.org
        Description . . . . . . . . . . . : Realtek RTL8139(A) PCI Fast Ethernet Adapter
        Physical Address. . . . . . . . . : 00-40-95-30-43-B4
        DHCP Enabled. . . . . . . . . . . : Yes
        Autoconfiguration Enabled . . . . : Yes
        IP Address. . . . . . . . . . . . : 192.168.1.99
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.2
        DHCP Server . . . . . . . . . . . : 192.168.1.2
        DNS Servers . . . . . . . . . . . : 168.95.1.1
                                            139.175.10.20
        Lease Obtained. . . . . . . . . . : 2002 年 11 月 24 日 AM 12:32:54
        Lease Expires . . . . . . . . . . : 2002 年 11 月 27 日 AM 12:32:54

C:\> ipconfig /renew  <==重新捉 IP 啦!
```

- 这样就 OK 啦！！很简单吧！

除错与检视租约档案：
接着下来，我们就要努力的来除虫了！除虫的第一步，就是察看登录档啦！最常发生的错误其实大概就是：

1. 写错字：不要笑！真的很容易写错字的！这很正常！所以大家要多多的去检查一下：
2. 没有加上『 ;』符号号：是的，这也是最常发生的错误啦！
3. 新版 DHCP 新增的功能限制：有时后也会有这样的讯息出现呢！

例如底下就是错误讯息的一般显示状态：

```
[root@test root]# vi /var/log/messages
Nov 23 23:32:25 vbird dhcpd: /etc/dhcpd.conf line 6: semicolon expected.
Nov 23 23:32:25 vbird dhcpd: option
Nov 23 23:32:25 vbird dhcpd: ^
Nov 23 23:32:25 vbird dhcpd: Configuration file errors encountered -- exiting
Nov 23 23:32:25 vbird dhcpd:
# 这一个例子告诉您，在第六行有错误！什么错误？呵呵！/etc/dhcpd.conf 中仔细检查啰！

Nov 23 23:34:12 vbird dhcpd: ** You must add a ddns-update-style statement to /etc/dhcpd.conf.
Nov 23 23:34:12 vbird dhcpd:    To get the same behaviour as in 3.0b2pl11 and previous
Nov 23 23:34:12 vbird dhcpd:    versions, add a line that says "ddns-update-style ad-hoc;"
Nov 23 23:34:12 vbird dhcpd:    Please read the dhcpd.conf manual page for more information. **
Nov 23 23:34:12 vbird dhcpd:
# 这个例子则在告诉您，您必须要新增一行字喔！不然不给您启动！ ^_^
```

这样就能够检验成功了！修改上面是很容易的啦！
再来要讨论的则是那个租约档案的内容问题！去看一下 /var/lib/dhcp/dhcpd.lease 的内容吧！

```
[root@test root]# vi /var/lib/dhcp/dhcpd.lease
lease 192.168.1.100 {
  starts 6 2002/11/23 16:15:22;
  ends 5 2002/11/29 16:15:22;
  tstp 5 2002/11/29 16:15:22;
  binding state active;
  next binding state free;
  hardware ethernet 00:90:cc:08:49:13;
  uid "\001\000\220\314\010I\023";
}
lease 192.168.1.99 {
  starts 6 2002/11/23 16:33:16;
  ends 2 2002/11/26 16:33:16;
  binding state active;
  next binding state free;
  hardware ethernet 00:40:95:30:43:b4;
  uid "\001\000@\2250C\264";
  client-hostname "tools";
}
lease 192.168.1.99 {
  starts 6 2002/11/23 16:33:21;
  ends 2 2002/11/26 16:33:21;
  binding state active;
  next binding state free;
  hardware ethernet 00:40:95:30:43:b4;
```

```
  uid "\001\000@\2250C\264";
  client-hostname "tools";
}
```

看到了吧！这个就是租约档案的内容啦！详细的记载何时申请的租约，以及期限在哪儿！嗯！这样就完成记录啰！

重点回顾

- DHCP (Dynamic Host Configuration Protocol) 可以提供网络参数给客户端 (client) 计算机自动设定其网络的功能；
- 透过 DHCP 的统一管理，在同一网域当中就比较不容易出现 IP 冲突的情况发生；
- DHCP 可以透过 MAC 的比对，来提供 Static IP (或称为静态 IP)，否则，通常提供客户端 dynamic IP (或称为动态 IP)；
- DHCP 除了 Static IP 与 Dynamic IP 之外，还可以提供租约行为之设定；
- 客户端离线、不明原因的当机、超过租约期限等机会下，DHCP Server 与客户端的租约行为会终止！
- DHCP 可以提供的 MAC 比对、Dynamic IP 的 IP 范围以及租约期限等等，都在 dhcpd.conf 这个档案当中设定的；
- 一般的情况之下，使用者需要自行设定 dhcpd.leases 这个档案，不过，真正的租约档案记录是在 /var/lib/dhcp/dhcpd.leases 里面；
- 在新版的 DHCP 的设定档 dhcpd.conf 当中，有时候需要加入『ddns-update-style ad-hoc;』才会正常的启动；

参考资源：

- Linux Magazine：http://www.linux-mag.com/2000-04/networknirvana_01.html
- DHCP mini HOWTO：http://www.tldp.org/HOWTO/mini/DHCP/index.html
- Internet Software consortium：http://www.isc.org/products/DHCP/
- Study Area：http://www.study-area.org/linux/servers/linux_dhcp.htm

---

本章习题练习 (要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- DHCP 的主要用途为何？
- DHCP 主要的两种 IP 分配模式为何？
- 在有 DHCP 主机存在的网域当中，且 client 端亦使用 DHCP 来规划客户端的网络参数，那么请问，在该网域当中，Client 端是如何取得 IP 的呢？？
- DHCP 是如何发送 Static IP 的？可以使用何种指令取得该信息？
- 在 DHCP 的租约档，亦即 /var/lib/dhcp/dhcpd.leases 当中，记录了什么信息？
- DHCP 的登录档放置于何处？

简易 DNS 服务器设定

最近更新日期：2004/10/31

我们知道计算机网络系统只认识所谓的 IP ，但是，您可能记得住网络上面所有主机的 IP 吗？ 就鸟哥来说，连自己的主机的 IP 都记不起来了，怎么可能连其它的主机 IP 都记的住！ 因为，人脑对于数字组成的 IP 的记忆实在是......不怎么样。 但是，相对来说，人们对于由文字所组成的主机名称那可以容易记忆的多了～ 所以，才会发展出可以经由主机名称(hostname)对应到计算机 IP 的一个模式，这样我们就可以轻轻松松的记住主机名称即可，计算机 IP 那就交给 Domain Name System (DNS)去搞定吧！

那个 DNS 系统是由柏克莱大学发展的 bind 这个套件(Berkeley Internet Name Domain)所提供的啦！基本上，DNS 最主要的工作就是将 Hostname 对应到 IP 这个功能了，不过，要架设一个成功的 DNS 主机的话，还得要对于 DNS 的运作很清楚才行啊！否则架设的不对，还反而会造成大家的问题喔！这个章节当中， 要学会的数据其实还蛮多的，需要了解：什么是正解、什么是反解、什么是 Zone、客户端 (Client) 是经由什么咚咚来查询得到 IP 的呢？！以及 DNS 的授权问题等等。 哇！赶快清一清脑门，要好好的用功啰！ ^_^

原理部分

目前人类的计算机网络里面，使用最普及的是所谓的 IP (IPv4)协议，透过这个协议， 我们可以将数据传送到任何一个可以连上 Internet 的地方。 不过，这个 IP 协议所设定的 IP 是由 32 个位，而转成十进制的话，是由 4 组数字所集合而成的， 例如 123.234.56.78 这样的格式。当我们利用 Internet 传送数据的时候， 就需要这个 IP ，否则数据怎么知道要被送到哪里去？(当然啦， 传送数据的方法有很多，不见得全部都是透过 IPv4 这个协议，例如 NetBIOS 就是一例。 不过，在这里，我们不讨论其它的传输方法，专门探讨 IPv4 这个协议喔！)

然而人脑对于 IP 这种数字的玩意儿，记忆力实在是不怎么样。但是如上所说， 当我们需要数据传输时，又很需要对方的 IP ，怎么办？为了应付这个问题， 早期的朋友想到一个方法，那就是利用某些特定的档案将主机名称与 IP 作一个对应， 让主机名称与 IP 有关连性，如此一来，我们就可以透过主机名称来取得该主机的 IP 了！ 真是个好主意，因为人类对于名字的记忆力可就好多了！ ^_^y。

可惜的是，该方法还是有缺憾的，那就是主机名称与 IP 的对应无法自动于所有的计算机内更新。 也就是说，我们必须要手动去所有的计算机里面更新该信息～天呐！哪有这么多时间～～ 而为了填补这个缺憾，柏克莱大学发展出另外一套阶层式管理主机名称对应 IP 的系统， 我们称他为 Berkeley Internet Name Domain, BIND ，这个系统可就优秀的多了～ 透过阶层式管理，可以轻松的进行维护的工作～太棒了！ 这也是目前全世界使用最广泛的领域名称系统(Domain Name System, DNS)哩～ 透过他，我们不需要知道主机的 IP ，只要知道该主机的名称，就能够轻易的连上该主机了！(在底下的说明当中，我们有时会提到 DNS 有时会提到 BIND ，这有什么不同？ 由上面的说明里面，您可以了解到， DNS 是一种因特网的协议名称，至于 Bind 则是提供这个 DNS 服务的套件～这样您了解了吗？！)

那么要立刻来架设 DNS 主机吗？当然不是～如同上面说的，因特网上面， 数据的传输最重要的就是得要知道对方的 IP，如此才能达成联机。因此， 架设 DNS 就必须要了解整体因特网的领域名称架构，否则，一旦 DNS 架设错误， 可能会造成您所管辖的主机无法正确的在 Internet 上头传输数据的问题！

所以，要设定 DNS 之前，您必须要就领域名称系统里面惯用的 FQDN、Hostname 与 IP 的查询流程， 正解与反解、合法授权的 DNS 主机之意义，以及 Zone 等等的知识作一个认识才行！ 这可是很重要的，不要轻忽他了！

什么是 Domain Name System：

DNS 的全名是『 Domain name system 』， 中文译名为『领域名称系统』， 这个咚咚的用途是什么哇！为什么我们的计算机或者是 Internet 一定需要他 (尤其是以 WWW 的方式来上网时) ？呵呵！ 他最大的用途就是『造福懒惰与记忆性薄弱的人类～』 哈哈！没错！为什么说他是造福人类呢？且听我娓娓道来：

- /etc/hosts 的历史：
  还记得我们在前几章当中提过的 网络基础 里面吧？目前在 Internet 上面通用的通讯协议为 TCP/IP ，那么数据传送是以 TCP 封包来传送，他还是建构在 IP 协议之上的，而众所皆知的，IP 是由四组 8 bit 的数字组成的，也就是类似『 xxx.yyy.zzz.www 』这样的型态，好啦，那么如果我们要连上某一部计算机， 就要在网址列输入该计算机主机的 IP 才能连接的上，如果是一部或两部计算机那还无所谓，如果像目前这种 Internet 的主机数目.....嘿!谁记得住这么多的 IP 呀！？

由于 IP 是一堆数字所组成的，实在不容易被懒惰与记忆性薄弱的人类所接受 (说的是鸟哥自己 ....)，那如果将这些数字以『名字』来取代呢？那又如何？ 也就是说， 只要输入一个『计算机的名字』而我们的系统就会自动的将这个名字转成计算机了解的 IP ！嘿嘿！如此一来，我要记得『名字』总是比 IP 容易的多了！

早期的人类早就想到这个简单又偷懒的方式了，那就是 /etc/hosts 这个档案的由来！例如，只要您输入 『 ping -c 5 localhost 』 您的 Linux 马上可以印出 127.0.0.1 这个 IP ， Why ？您去看一下 /etc/hosts 就知道为什么了！所以啰，只要将您常常上网的『网址对应的 IP 』写到这个 /etc/hosts 底下，您的 IP 搜寻速度就会快上很多～(注：再次强调，在您的私有网域内部，最好将所有的 IP 都写入这个档案中啦！)

- DNS 的历史：
  早期(大约 20~30 年前)的计算机可是贵重物资，一般人是可望而不可及的，因为计算机数量太少，所以使用 /etc/hosts 来记忆这些 IP 与名称的对应还尚可应付。但是在现代， Internet 上面这么多主机，并且常常会突然的『噗通』又多出一部主机来服务， 那么我们总不能一个一个的将他输入在 /etc/hosts 里面吧！？对呀！真不聪明！ 所以后来的这个时候就有所谓的『领域名称解析系统, DNS』出现啦！

  DNS 利用类似树状目录的型态，将主机名称的管理分配在不同层级的 DNS 主机当中，经由分层管理，所以每一部主机的记忆的信息就不会很多， 而且异动上面也相当的容易修改！那么这个 DNS 的功能您知道了吗？对啦！就是『将计算机主机的名称转译成 IP 』就是了！当然啰，他的额外功能还很多！总之，他的最大功能就是『 让有意义的，人类较容易记忆的主机名称(英文字母)，转译成为计算机所熟悉的 IP 地址！ 』

  举个例子来说好了，奇摩雅虎的网站的 IP 是 202.1.237.21 ，所以您可以在您的浏览器上面输入『 http://202.1.237.21 』来连上奇摩雅虎！ 不过，我想没有几个人能够将这个 IP 背的起来的吧？！反之，我们却都知道奇摩雅虎的网址为 tw.yahoo.com ，那么您只要输入『 http://tw.yahoo.com 』就可以连上 Internet 啦！很容易记忆吧！

```
/etc/hosts ：直接在 Client 端的档案内输入主机名称对应的 IP 来查询；
DNS 系统  ：利用额外的 DNS 服务，让 Client 端可以透过名称解析来取得目的地主机的 IP
```

-
- Fully Qualified Domain Name (FQDN)
  在提到名称与 IP 的解析流程之前，我们还必需来讨论一下『什么是 domain name 与 host name ？』也就是，领域名称与主机名称。 在讨论这个主题之前，我们来聊一聊比较生活化的话题，请注意喔！ 底下的例子不涉及政治！呵呵！先提一下，免得大家敏感：

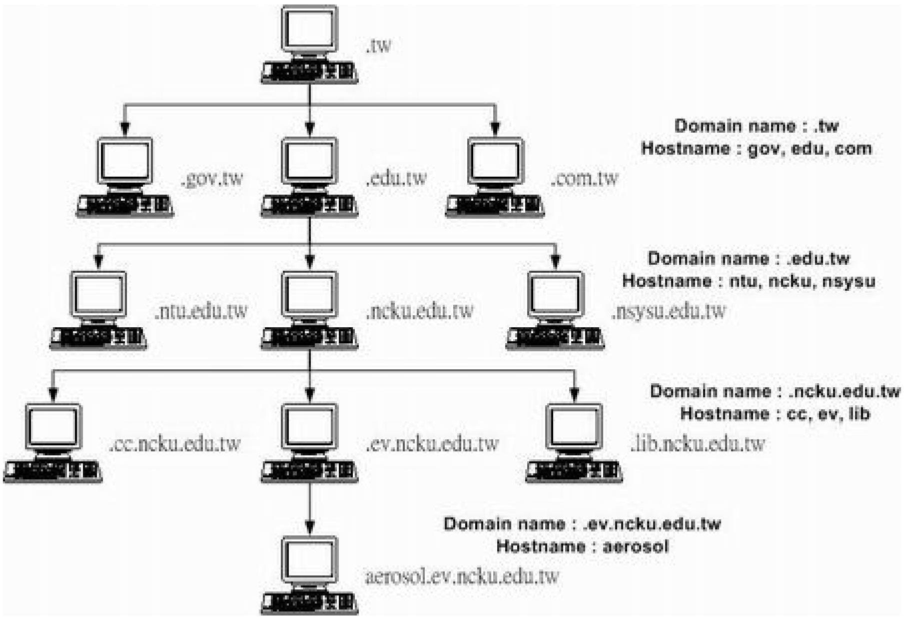
  - 我们晓得全台湾有很多个『李登辉』，这个『李登辉』就代表每一个独立的个人！ 但是您怎么知道这个李登辉跟前总统李登辉是否为同一个人？咦！ 每个李登辉都来自不同的县市嘛！对啦，所以我们就以县市来做为区分， 所以有台北的李登辉跟高雄的李登辉，这两个就可以分辨了！嘎！万一不幸， 台北还有两个李登辉怎么办？那就用乡镇来分呀！所以有台北、三芝的李登辉跟台北、 仁爱的李登辉，如果我们将他列出来，可以这样看：

```
李登辉、三芝、台北
李登辉、仁爱、台北
李登辉、高雄
....
```

- 是否就可以分辨他的不同点了呢？呵呵！没错！就是这样！
- 另外一个例子可以使用电话号码来看，假如高雄有个 1234567 而台南也有个 1234567 ，那么(1)您在高雄直接拨接 1234567 时，他会直接挂入高雄的 1234567 电话中， (2) 但如果您要拨到台南去，就得加入(06)这个区码才行！我们就是使用区码来做为辨识之用的！

是否还不清楚我要说什么？呵呵！我们常常会发现主机名称都是 www 的网站，例如 www.gov.tw, www.seednet.net, www.hinet.net 等等，那么我们怎么知道这些 www 名称的主机在不同的地方呢？就需要给他领域名称啰！也就是 gov.tw, seednet.net, hinet.net 等等的不同，所以即使您的主机名称相同，但是只要不是在同一个领域内， 那么就可以被接受啰！

基本上，我们知道 DNS 是有层级之分的，那么每个层级的 Hostname 与 Domain name 可是不一样的咚咚ㄋㄟ～我们可以使用我们的主机来加以说明，如下图所示：



图一、分层次的 DNS 架构 (Hostname 与 Domain name)

在上面的例子当中，由上向下数的第二层里面，那个 .tw 是 domain name ，而 com, edu, gov 则是主机的名称，而在这个主机的名称之管理下，还有其它更小网域的主机， 所以在第三层的时候，基本上，那个 edu.tw 就变成了 domain name 了！而成大与中山的 ncku, nsysu 则成为了 hostname 啰！

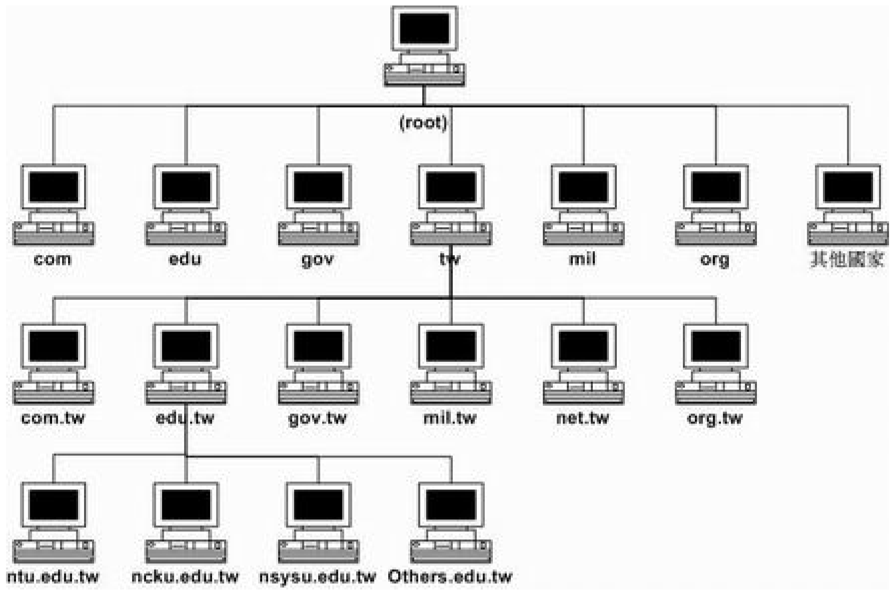
呵呵！以此类推，最后得到我们的主机那个 aerosol 是主机名称，而 domain name 是由

ev.ncku.edu.tw 那个名字所决定的!自然,我们的主机就是让管理 ev.ncku.edu.tw 这个 domain name 的 DNS 主机所管理的啰! 这样是否了解了 domain name 与 hostname 的不同了呢?

DNS 的查询过程:

接下来我们要谈一谈,那么 DNS 的 (1)架构是怎样? (2)查询原理是怎样? 总是要先知道架构才能知道如何查询的呐! 所以底下我们先来介绍一下整体的架构。

- DNS 的架构:



图二、DNS 层阶概念示意图

上面就是一个简单的 DNS 阶层架构啰, 最上方一定是 . (小数点) 这个 root 的 DNS 主机, 他底下管理的就只有 com, edu, gov, mil, org 与以国家为分类的第二层的主机名称了! 例如台湾地区最上层的领域名称是以 .tw 为开头, 管理这个领域名称的这部机器的 IP 是在台湾, 但是他的记录则是记录在 . (root)那部机器里面的!还有其它的国家的最上层如 .cn 指的是大陆, .de 指的是德国一样!那么每个国家之下记录的主要的下层有哪些领域呢? 呵呵!主要就是有这六大类:

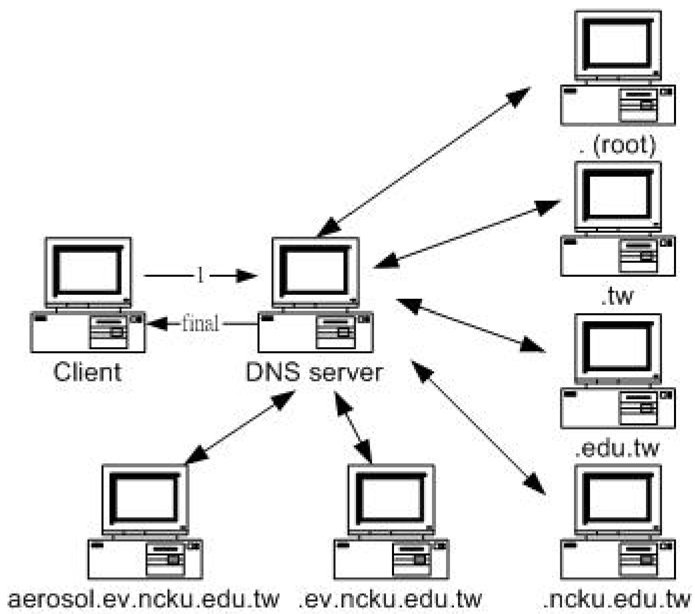
名称	代表意义
com	公司、行号、企业
org	组织、机构
edu	教育单位
gov	政府单位
net	网络、通讯
mil	军事单位

其实最早之前在 . (root)之下只有这六大类的 domain name ，但是网络成长的速度太快了，因此后来又多出这些以国码来分的 domain name ，如此一来，在该国家之内，只要向该国家申请 domain name 即可，不需要再到最上层去申请啰！也因此，在这些国码之下，还是有这六大类的 domain name 为主的哩！当然啦，在目前，由于因特网持续的发烧，说实在的 domain name 实在是有点不太够用,所以又有相当多的领域名称被设计出来,例如目前台湾 ISP 提供的 .idv.tw 的个人网站啦！

好了，再强调一次， DNS 系统是以所谓的阶层式的管理，所以，请注意喔！那个 .tw 只记录底下那一层的这六个主要的 domain 的主机而已!至于例如 edu.tw 底下还有个 ncku.edu.tw 这部机器，那就直接授权交给 edu.tw 那部机器去管理了！也就是说『 每个上一层的 DNS 主机，所记录的信息，其实只有其下一层的主机名称而已！ 』至于再下一层，则直接『授权』给下层的某部主机来管理啰！呵呵！所以您就应该会知道 DNS 到底是如何管理的吧！ ^_^

会这样设定的原因不是没有道理的！ 这样设计的好处就是：每部机器管理的只有下一层的 hostname 对应 IP 而已，所以减少了管理上的困扰！而下层 Client 端如果有问题，只要询问上一层的 DNS server 即可！不需要跨越上层，除错上面也会比较简单呢！

- DNS 的搜寻流程：
  刚刚说过 DNS 是以类似『树状目录』 的型态来进行名称的管理的！所以每一部 DNS 主机都『仅管理下一层 DNS 主机的名称转译』而已， 至于下层的下层，则『授权』给下层的 DNS 主机来管理啦！这样说好像很绕口，好吧！我们就以下图来说一说原理啰：



图三、DNS 主机查询流程示意图

首先，当您在浏览器的网址列输入 http://aerosol.ev.ncku.edu.tw 时，您的计算机就会依据相关设定(在 Linux 底下就是利用 /etc/resolv.conf 这个档案) 所提供的 DNS 的 IP 去进行联机查询，好了，由于目前最常见的 DNS 主机就属 Hinet 的 168.95.1.1 这个 DNS 了，所以我们就拿他来做例子吧！嗯！这个时候， hinet 的这部主机会这样工作：

1. 先查看本身有没有纪录：
   刚刚说过啦，由于 DNS 是层阶式的架构，任何一部 DNS 都仅记录下一层里面的主机名称对应的 IP 而已，由于 hinet 并非学术网络里面的主机，所以自然也就没有办法直接提供给 client 端关于 aerosol.ev.ncku.edu.tw 这部机器的 IP 了，所以啦，一般而言，这个时候 168.95.1.1 就会向最顶层，也就是 .(root) 的主机查询 .tw 这部机器的地址；
2. 向最顶层 (root)查询：
   由于 168.95.1.1 没有纪录我们主机的 IP ，这个时候他就会向『最顶层』的 .(root) 这部主机来查询 .(root) 的下一层，也就是 .tw 这部机器的数据了！这个时候， .(root) 就会告诉 168.95.1.1 说『嘿！您要查 .tw 这个网域的管理者呀！？喝！我这里有 .tw 这个网域的管理的主机之 IP 信息，您可以直接去找他！』；
3. 向第二层查询：
   168.95.1.1 接着又到 .tw 去查询，而该部机器管理的又仅有 .edu.tw, .com.tw, gov.tw... 那几部主机，经过比对后发现我们要的是 .edu.tw 的网域，所以这个时候 .tw 又告诉 168.95.1.1 说：『您要去管理 .edu.tw 这个网域的主机那里查询，我有他的 IP ！』；
4. 向下层持续查询：
   好了，一步一步下来， .edu.tw 可以查到管理 .ncku.edu.tw 的主机 IP ； .ncku.edu.tw 可以查到管理 .ev.ncku.edu.tw 的主机 IP ，而最后我们 aerosol.ev.ncku.edu.tw 就可在管理 .ev.ncku.edu.tw 网域的那部主机的设定纪录当中查询到啦！
5. 记录暂存内存：
   查到了 IP 之后，这部 168.95.1.1 的 DNS 机器总不会在下次有人查询 aerosol.ev.ncku.edu.tw 的时候再跑一次这样的流程吧！粉远粉累的呐！而且也很耗系统的资源与网络的频宽，所以呢， 168.95.1.1 这个 DNS 很聪明的会先记录一份 aerosol.ev.ncku.edu.tw 对应 IP 的信息在自己的暂存内存当中，以方便下一次又有人对同一个主机名称的要求之查询！最后则将结果回报给 client 端！当然啦，那个记忆在 cache 当中的数据，其实是有时间性的，当过了 DNS 设定记忆的时间(通常可能是 24 小时)，那么该记录就会被释放喔！

由这样的分层负责您发现了什么？嗯！那就是：

- 当一个『合法』的 DNS 主机里面的设定修改了之后，来自世界各地任何一个 DNS 的要求，都会正确无误的显示正确的主机名称对应 IP 的信息，因为他们会一层一层的寻找下来，所以，要找您的主机名称对应的 IP 就一定得要透过您的上层 DNS 主机的纪录才行！所以只要您的主机名字是经过上层『合法的 DNS』主机的设定的，那么就可以在 Internet 上面被查询到啦！呵呵！很简单维护吧，机动性也很高。
- 在主机的暂存内存记录当中，由于是有时间性的，所以当您的主机名称在 DNS 当中被修改了之后，但是由于之前的旧信息还记忆在其它的 DNS 主机的暂存内存里面，所以啦，可能在别人以非您的 DNS 主机来查询您的主机名称时，就会得到先前的旧信息，这个时间差不多可能是 10 分钟到 2 天左右，这也是为什么我们常说当您修改了一个 domain name 之后，可能要 2 ~ 3 天后才能全面的启用的缘故啦！

好啦！哇！既然 DNS 这么棒，然后我们又需要架站，所以需要一个主机的名称，因此， 那么我们需要架设 DNS 了吗？！哈哈！当然不是，为什么呢？ 刚刚鸟哥提到了很多次的『合法』的字眼，因为他就牵涉到『授权』的问题了！ 我们在前面的『申请合法的主机名称』当中也提到， 只要主机名称合法即可，不见得需要架设 DNS 的啦！

- DNS 使用的 port number ：
  好了，既然 DNS 系统使用的是网络的查询，那么自然需要有开 Listen 的 port 啰（ 监听的埠号 ）！没错！很合理！那么 DNS 使用的是那一个 port 呢？那就是 53 这个 port 啦！您可以到您的 Linux 底下的 /etc/services 这个档案看看！搜寻一下 domain 这个关键词，就可以查到 53 这个 port 啦！但是这里需要跟大家报告的是，通常， DNS 查询的时候，是以 udp 这个较快速的数据传输协议 (protocol) 来查询的，但是万一没有办法查询到完整的信息时，就会再次的以 TCP 这个协定来重新查询的！所以启动 DNS 的 daemon (就是 named 啦) 时，会同时启动 TCP 及 udp 的 53 这个 port number 喔！

关于『授权』的意义：

很多朋友都认为『 架设 DNS 可以设定主机的名称，而我要架站需要主机有名字，因此一定需要架设 DNS ，只要有 DNS，我的主机就可以有名字了！ 』是这样吗？当然不是！这是错误的观念！怎么说呢？

从上面图三的图示当中，您应该不难发现，当我要搜寻 aerosol.ev.ncku.edu.tw 主机时，就需要向管理 .ev.ncku.edu.tw 这个网域的那部机器查询才行，而要查询 .ev.ncku.edu.tw 则需要在 .ncku.edu.tw 上面询问才可以!这是因为『上层 DNS 主机 .ncku.edu.tw 已经将 .ev.ncku.edu.tw 这个网域的管理权 " 授权" 给 green.ev.ncku.edu.tw 这部机器，当有人要查询 .ev.ncku.edu.tw 这个网域的主机 IP 时， .ncku.edu.tw 将会把查询的任务直接转给 green.ev.ncku.edu.tw 去管理了！从此， .ncku.edu.tw 这个网域的管理主机，将不会再接管 ev.ncku.edu.tw 这个网域的名称管理！ 』是否很像人类社会的『授权』的概念？

也就是说，当您老板充分的『授权』给您某项工作的时候，从此， 要进行该项工作的任何人，从老板那边知道您才是真正『有权』的人之后，都必须要向您请示一样！ ^_^！所以啰，如果您要架设 DNS ，而且是可以连上 Internet 上面的 DNS 时，您就必须要透过『上层 DNS 主机的授权』才行！这是很重要的观念喔！等一下我们在底下会介绍一个如何架设一个 『经过合法授权的 DNS 主机』哩！

其实，如果将上面的话改换成：『 我要架站，所以我要让我的主机有一个合法的名字！ 』那样就合理了！怎么说呢？因为我可以请上层 DNS 帮我设定主机名称对应 IP 就可以啦！如此一来，要找我的 hostname 对应 IP 的人，都可以直接在我的上层 DNS 里面找到，根本不需要透过我的 Linux 主机呐！例如鸟哥研究室的 aerosol.ev.ncku.edu.tw 就可以在 green.ev.ncku.edu.tw 这部管理 DNS 的 server 上面找到ㄋㄟ～不必亲自来我的 aerosol.ev.ncku.edu.tw 上面找！也就是说，藉由 DNS 系统最大的功能『主机名称转译成 IP 』 这个动作，那么您只要向任何一个合法的 DNS 主机申请一个『主机名称, hostname 』给您的 Linux 主机，让大家都可以藉由该 DNS 主机来查询到您的 Linux 之 IP ，就可以使用该主机名称来架站啦！就是这么简单！

好了，那么您就应该知道了，要让您的主机名称对应 IP 且让 Internet 上面的计算机都可以查询的到，就需要：

1. 上层 DNS 的授权让您设定 DNS 主机，或者是；
2. 直接请上层 DNS 主机来帮您设定！

这两种模式，那么哪种模式比较好呢？这没有一定的答案， 底下我们来谈一谈，您比较适合哪一种模式的设定呢？

网站代管还是自己设定 DNS：

如果您曾经申请过 domain name 的话，例如向 Hinet 或 Seednet 等台湾各大主要 ISP 申请 domain name 的话，应该都会知道有两种主要的模式，就是刚刚上头提到的 DNS 授权，或者是直接交给 ISP 来管理。交给 ISP 管理的， 就可以称作是网站代管啦！当然啦，如果您是学校单位的话，或者是企业内部的小单位，那么就得请您向上层 DNS 主机的负责人要求啰！无论如何，您只能有两个选择就是了，要不就是请他帮忙您设定好 hostname 对应 IP ，要嘛就是请他直接将某个 domain name 段授权给您做为 DNS 的主要管理网域。那么我怎么知道那个方式对我比较好呢？请注意，由于 DNS 架设之后，会多出一个监听的 port ，所以理论上，是比较不安全的！因此，能不设当然就不要设定比较好啰！所以，这里的建议如下：

- 需要架设 DNS 的时机：
    - 您所负责需要连上 Internet 的主机数量庞大：例如您一个人负责整个公司十几部的网络 Server ，而这些 Server 都是挂载您的公司网域之下的。这个时候想要不架设 DNS 也粉难啦！
    - 您可能需要时常的修改您的 Server 的名字，或者是您的 Server 有随时增加的可能性与变动性；

- 不需要架设 DNS 的时机：
    - 网络主机数量很少：例如家里或公司只有需要一部 mail server 时；
    - 您可以直接请上层 DNS 主机管理员帮您设定好 Hostname 的对应时；
    - 您对于 DNS 的认知不足时，如果架设反而容易造成网络不通的情况；
    - 架设 DNS 的费用很高时！

正解与反解的 Zone 意义：

讲了这许多，还得再提一提关于正解、反解与 Zone 的问题才行啊！

- 什么是正解与反解？
  我们在前头的开宗明义当中就提到啦， DNS 系统本来最主要的功能就是在转译 hostname 与 IP 啰，由于计算机在网络上面其实认识的只是 IP 啦，所以，一般来说，我们称『由 hostname 去寻找出 IP 的程序称为 正解 』， 至于由 IP 去查询得到 hostname 那就被称为反解了！正反解的设定情况是差异性很大的！怎么说呢？

    - 正解：
      在正解的情况之下，我们可以透过主机分层设定的方式来查询(例如前面的图三)，而因

为是 Hostname 对应 IP ，所以即使在不同网段的 IP ，仍然可以写在同一个 domain 之中！例如我的主机是在学校里面（ 140.116.xxx.xxx ），但是我申请的是 vbird.idv.tw 这个 domain 的名称，而很多朋友则是以 ISP 提供的 IP（ 例如 61.xxx.xxx.xxx ） 来进行 *.idv.tw 的申请的！呵呵！那么一来，我的 vbird.idv.tw 就与大家的 *.idv.tw 在同一个 domain 的设定当中啰，但是这些主机却是在不同的网域之中喔 (140.116.xxx.xxx 不会跟 61.xxx.xxx.xxx 在同一个网段中吧！ ^_^）！所以啰，任何一部 DNS 都可以将您的 IP 写入他们的正解当中啰！

- 反解：
  但是反之则不行！怎么说呢？因为当初 IP 规划分配的时候，就必需要一个区域一个区域的划分的，所以当然不可能同一个网段的 IP 在不同的地方出现吧！因为这涉及到 TCP/IP 的协议与 router 的架构ㄋㄟ～因此，同一个 IP 网段的反解就真的得要透过上层主机的设定才行了！所以由 IP 反查 hostname 的话，那么大部分的情况下，就需要向直属的上层申请了！

举个例子来说：我想要自己的领域名称的名字，所以我可以去外面的 ISP 申请注册一个合法的名字来架设我的 DNS ！从此之后，别人就可以经过我的 DNS 正解查询得到我的主机 IP。但是如果要由 IP 反查回 hostname 的话，我就『一定必需要』请管理我主机所在网域的上层的 DNS 管理员来设定才行ㄋㄟ！这也是目前比较麻烦的地方，因为正解您可以自行设定， 但是反解则必需要请上层的管理员设定！如果是向 ISP 申请的 IP ，那就得向 ISP 申请反解名称改换，这个部分通常很麻烦～

- 什么是 Zone ？
  知道正反解之后，再来要来知道一下，什么又是 Zone（ 区域 ） 呢？说的简单一点的话，一个正解或反解的设定就是一个 zone ，例如我要规范 vbird.idv.tw 这个 domain 的设定内容，那么他就是一个 zone ！通常，『一个设定档就是一个 zone 』！ 如果以鸟哥的 vbird.idv.tw 这个例子来说，配合上面的 图三 来说明， 那鸟哥的 vbird.idv.tw 那部主机里面至少需要知道 .(root) 以及鸟哥自身的设定， 所以，这个 domain 的 DNS 设定档里面，必需要有：

    - hint（ root ） 的设定；
    - vbird.idv.tw 这个 domain 的正解设定；
    - localhost 的正解设定(非必要)；
    - localhost 的反解设定(非必要)。

  那么我就有四个 zone 了！如果以我们系馆的 DNS 主机 green.ev.ncku.edu.tw 来说的话，他至少要有：

    - hint(root)；
    - ev.ncku.edu.tw 正解；
    - ev.ncku.edu.tw 反解以及 ；
    - localhost 正解；
    - localhost 反解。

等五个 zone 的定义啰！嘿！您会发现，我没有 vbird.idv.tw 这个 domain 的反解设定～为什么呢？请参考上面的说明吧！因为反解需要要求 IP 协议的上层来设定才行！并且，需要特别留意的是,『每一个 zone 都有一个设定档,而规定这些设定档档名的,就交给 /etc/named.conf 这个参数档来设定！』（在新版的 Linux distribution 当中， 也有将这个档案放置在 /var/named/chroot 当中喔！需要特别注意！)也就是说， DNS server 使用的 bind 这个套件中，他的主要参数档是 /etc/named.conf ，而这个档案当中就是记录了每一个 zone 的设定档档名！实际上，主机名称与 IP 的对应就是记录在各个 zone 的设定档内～

- 正反解一定要成套吗？
  好了，正反解需不需要成套产生，在这里不用多说明了吧！？ ^_^！请注意喔，在很多的情况下，尤其是目前好多莫名其妙的领域名称产生出来， 所以，常常会只有正解的设定需求而已。不过也不需要太过担心啦， 因为通常在反查的情况中，如果您是使用目前台湾地区最流行的 ADSL 上网的话，那么 ISP 早就已经帮您设定好反解了！例如：211.74.253.91 这个 seednet 的浮动式 IP 反查的结果会得到 91.253.74.211.in-addr.arpa 这样的主机名称！所以在一般我们自行申请领域名称的时候，您只要担心正解的设定即可！ 不然的话，反正反解的授权根本也不会开放给您，您自己设定得很高兴也没有用呀！ ^_^

架设 DNS 所需要的套件：

- 安装 DNS 套件： BIND
  终于废话都说完了！相信您大概也有点累的吧！？鸟哥是蛮累的啦，因为手臂、 肩颈酸痛的毛病颇严重....咦！讲这个干嘛！？ @_@ 好啦，我们终于要来安装 DNS 所需要的套件了！还记得前面提过的，我们要使用的 DNS 就是使用柏克莱大学发展出来的 BIND（Berkeley Internet Name Domain, BIND）这个套件啦！那么怎么知道您安装了没？还记得 基础篇 里面的 RPM 吗？对啦！就是使用 RPM 来检验啰：

```
[root@test root]# rpm -qa | grep bind
bind-9.2.1-4mdk                <==这个是用来安装 Server 的
bind-utils-9.2.1-4mdk          <==这个是用来做为 Client 端搜寻 domain name 的指令
```

- 万一没装怎么办？嗄！还问我ㄞ～赶快将您的原版光盘拿出来，然后将他安装上去先～不会安装？ 请自行拿出『鸟哥的 Linux 私房菜 -- 基础学习篇』去观察一下 RPM 的用法吧！
- BIND 的预设路径设定：
  基本上， BIND 的主要设定档为 /etc/named.conf 这一支档案，各种针对主机的设定值都在这个档案中设定的！但是对于 hostname <--> IP 的对应关系，就需要由 zone 来设定了！但是这个 zone 的文件名称是在 /etc/named.conf 里面规定的！所以，请注意喔，每一个 zone 的名称都是可变的，但是需要在 /etc/named.conf 里面命名好！此外，最好将 zone 规定出来的档案直接放置到 BIND 的预设 zone 摆放的目录，就是 /var/named 里面去！比较好管理啰！
- BIND 的升级：
  必须请大家注意的是，这个 DNS 的 53 port 其实也不是个很安全的咚咚，所以呢，非必要，其

实是不太建议启用 DNS 的啦！不过，如果真的要安装的话，那么请随时注意您的 Linux distribution 是否有定时的公告的漏洞修补套件呢？这个真的很重要，因为鸟哥很久很久以前，就是被这个 port 53 给种植了一个蠕虫，真是讨厌的很！ @_@

设定部分：

在 DNS 的设定上面，基本上，您必须要已经很清楚 zone 是什么了，否则很难继续设定喔！会搞的一塌糊涂的～无论如何，您一定要知道的是， bind 的设定档，就是 /etc/named.conf （ 在新版的 BIND 当中，这个档案似乎已经被搬移到 /var/named/chroot 当中了！您必须自行寻找～ ） 这个档案，如果他不存在的话，请自行建立吧！另外，针对 DNS server 的类型大致上可以分为三类，分别是：

- Master：
  这种类型的 DNS 本身含有领域名称的设定档（ 就是有 Zone 啦！ ），这个设定档就是设定正解或者是反解的『Database』啰！所以他本身是具有提供 Internet 查询所需的数据喔！例如我可以在我的主机上面设定提供 vbird.idv.tw 这个网域，那么我的主机就是 master 类型的主机啦！
- Slave：
  既然想要架设 DNS 主机，自然就是希望自己的主机名称能够在 Internet 上面被查询的到。 而您也晓得，计算机主机这东西什么时候会挂点？网络环境这玩意儿，什么时候会死掉？ 是谁也不敢说的～而，您总不希望自己的主机名称无法被查询到吧？ 所以，一般来说， DNS 系统通常会建议您至少要有两部主机提供 DNS 的服务～

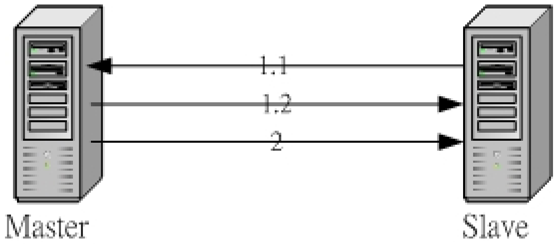
  不过，如果您有四部 DNS 主机提供这样的名称解析服务， 而且这四部是互相为备援的，也就是说，这四部主机的内容其实是一模一样的， 那么如果您要更动一个 IP 与名称的对应时，就必须要手动去修订四部主机的内容， 这样会不会很麻烦啊？！

  这个时候就有 slave 类型的 DNS 主机出现了！不过，slave 主机必须要与 master 主机相互搭配喔！ 以上面的案例来说明，如果我必须要有四部主机提供 DNS 服务，且四部内容相同， 那么我只要指定一部主机为 Master ，其它三部为该 Master 的 Slave 主机， 那么当要修改一部名称对应时，我只要手动更改 master 那部机器的设定档，然后， 重新启动 BIND 这个服务后，呵呵！其它三部 slave 就会自动的被通知更新了！ 这样一来，在维护上面可就轻松写意的多了～

  由于目前因特网并不安全，同样的 BIND 服务也不是很安全的～ 因此，如果您设定 Master/Slave 架构时，您的 Master 主机必须要限制 只有某些特定 IP 的主机能够取得您 DNS 主机的正反解数据库才好！ 所以，上面才会提到 Master/Slave 必须要互相搭配才行！

  另外，既然我的所有 DNS 主机是需要同时提供 internet 上面的领域名称解析的服务， 所以不论是 Master 还是 Slave 主机，他都必须要可以同时提供 DNS 的服务才好！ 因为在 DNS 系统当中，领域名称的查询是『先抢先赢』的状态，所以， 我们不会晓得哪一部主机的数据会先被查询到的！ 为了提供良好的 DNS 服务，每部 DNS 主机都要能正常工作才好啊！
- Cache-only：
  通常设定在防火墙上面的呢！这种类型的 DNS 主机没有自己的数据库，单纯仅帮助 Client 端向外部的 DNS 主机要求数据而已～简单的来说，他可以想成是一个『代理人』的角色而已～

那么 Master/Slave 的数据更新到底是如何动作的呢？请注意，Slave 是需要更新来自 Master 的 DNS 数据啊！所以当然 Slave 在设定之初就需要存在 Master 才行喔！好了， Master 与 Slave 的数据同步动作可以由底下的图示来看：



图四、Master/Slave 的 DNS 主机数据同步过程

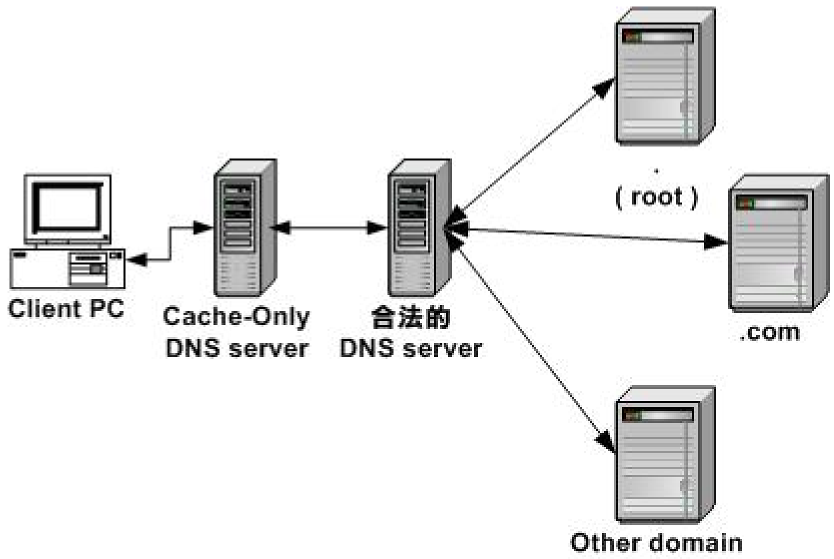
- 判断是否需要更新(1.1)：
  我们可以在 Slave 设定好向 Master DNS 主机要求数据更新的周期时间，则每当到达更新时间时， Slave 会向 Master 索取是否需要更新数据，这个更新数据的判断则以 Serial number 是否不同来进行更新喔！
- 判断是否需要更新(1.2)：
  除了由 Slave 向 Master 的查询之外，Master 如果 DNS 数据经过变更，且想要 Slave 同步更新时，也可以主动的向 Slave 进行更新通知！
- 数据同步化(2)：
  最后当然就是数据由 Master 传送到 Slave 来更新 Slave 的 DNS 数据啰！

请注意，如果您想要架设 Master/Slave 的 DNS 架构时，两部主机 (Master/Slave) 都需要您能够掌控才行！网络上很多的文件在这个地方都有点『闪失』， 请特别的留意啊！

底下我们就来谈一谈几个简单的 DNS 主机，分别是 cache-only (单纯 forward) 与较为详细的 Master 类型的 DNS 主机，最后，我们再以一个简易的 slave 主机设定来作为结尾～

单纯的 forward DNS 主机设定：

什么是单纯的 forward DNS 的主机呢？
好了，了解了 BIND 的预设路径之后，我们知道了 BIND 主要设定档是 /etc/named.conf 这个档案，但是偏偏我的 /etc 底下就没有这个档案！哈哈！因为您要自行建立啦！ ^_^！在介绍怎么设定每一个 zone 之前，我们先来玩一个简单的 DNS 主机！就是 cache-only DNS server ！ 也称为 forward DNS 啰！顾名思义，这个 DNS server 只有 cache (快取) 的功能，也就是说，他本身并没有主机名称与 IP 正反解的设定档， 完全是由对外的查询来提供他的数据来源！因为他没有 zone 的设定档，所以他就必须要连上一部合法的 DNS 才行！整个运作的流程可以看成是这个样子：



图五、Cache-Only DNS 主机的运作流程

由上面的图示来看，您可以发现，其实，我们 Client 端虽然都是使用 Cache-Only 的 DNS 在搜寻，但是，实际上 Cache-only 的主机都是请一个 (Forwarders) DNS 主机来帮忙查询的，本身并没有 zone 的设定档啦！所以说，基本上， cache-only 的 DNS 只是一个中间传递数据的 DNS 主机罢了！那么为什么要架设这样的一个 DNS 主机呢？闲闲没事干？当然不是！这是有原因的啦！底下说给您听啰！

什么时候使用 cache-only DNS?
在某些公司行号里头，为了预防员工利用公司的网络资源作自己的事情，所以， 都会针对 Internet 的联机作比较严格的限制。当然啦，连 port 53 这个 DNS 会用到的 port 也可能会被挡在防火墙之外的～这个时候， 您可以在『防火墙的那部机器上面，加装一个 cache-only 的 DNS 服务！』这是什么意思呢？很简单啊！就是您自己利用自己的 防火墙主机上的 DNS 服务去帮您的 Client 端解译 hostname <--> IP 啰!因为防火墙主机 可以设定放行自己的 DNS 功能,而 Client 端就设定该防火墙 IP 为 DNS 主机的 IP 即可！ 哈哈！这样就可以取得主机名称与 IP 的转译啦！

简易的 cache-only DNS 设定：
设定一个 cache-only 的 DNS 主机其实真的很简单的啦!因为不需要设定正反解的 Zone ，所以只要设定一个档案(就是 named.conf)即可！真是快乐得不得了呐!

---

1. 编辑 /etc/named.conf
   在这个档案中，主要是定义跟主机有关的事项，以及各个 Zone 的代表含意与档案，因为 cache-only 没有 Zone ，所以我们只要设定好跟主机有关的设定即可。设定这个档案的时候请注意：

   - 批注数据是以『 // 』来作设定的!
   - 每个段落之后都需要以『 ; 』来做为结尾!

   那么您可以这样设定这个档案啦!

```
[root@test root]# vi /etc/named.conf

// This settings is only for forwarding DNS Server
options {
        pid-file "/var/run/named/named.pid";
        forward only;            //只允许 forward!
        forwarders {
                168.95.1.1;      //我这里使用 hinet 的 DNS !
                139.175.10.20;   //这个是 seednet 的 DNS !
        };
};
// 我这里有设定 pid-file ，所以得要特别注意了！因为 pid-file
// 所在的目录下，也就是那个 /var/run/named 目录，我的 bind
// 执行文件程序的拥有人( 正常应该是 name 这个使用者 )必须要
// 能够写入！也就是说， /var/run/named 的 owner 必须
// 是 named 这个 user 才行喔！

[root@test root]# ls -al /var/run/named
total 12
drwxr-xr-x    2 named    named        4096 Dec  5 02:28 ./
drwxr-xr-x   10 root     root         4096 Dec  5 02:01 ../
# 注意上面那个 ./ 目录的拥有者！
```

我们仅动用到 option 这个参数而已，里面的设定值意义为：

- pid-file
  指的是每一个 services 的记录自己的 PID (Process ID) 的档案啰！这个档案通常用在重新启动或者是 reload 整个 services 最常被使用到的！因为可以使用 kill -1 PID 来重新启动啊！嗄！忘记什么是 PID ？赶快拿出基础篇复习一下！
- forwarders
  (不要忘记那个 s 喔！)就是要设定往前寻找的那个『合法』的 DNS 啰！每一个 forward 的主机之 IP 都需要有『 ; 』来做为结尾！
- forward only
  这个设定可以让您的 DNS 主机仅进行 forward 而已！是 Cache-Only 主机最常见的设定了！

很简单吧！这样就已经设定完成了最简单的 cache-only 的 DNS 主机了！

---

2. 启动 named
   启动总不会忘记吧！？赶快去启动一下吧！

```
[root@test root]# /etc/rc.d/init.d/named start
Starting named:                                         [  OK  ]
```

3.

4. ---

5. 观察 port 的变化

   请特别的注意喔！并不是启动的时候显示 OK 就会成功的！所以，还要赶快的来看一下您的 port 53 有没有启动ㄋㄟ～

```
[root@test root]# netstat -utln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address        Foreign Address      State
tcp        0      0 192.168.1.2:53       0.0.0.0:*            LISTEN
tcp        0      0 127.0.0.1:53         0.0.0.0:*            LISTEN
udp        0      0 192.168.1.2:53       0.0.0.0:*
udp        0      0 127.0.0.1:53         0.0.0.0:*
```

   特别需要留意的是，如果没有指定接口的话，那么所有的网络接口，包含 lo, eth0, ... 等接口都会被设定为可以接受 domain name 要求的响应接口！此外， 还记得我们在前面提到的，每个接口同时都会提供 TCP 与 UDP 封包的服务喔！ 这样看起来似乎真的有启动的样子，不过，我们还是得瞧一瞧设定方面有没有什么大问题呢？

---

6. 检查 /var/log/messages 的内容讯息

   named 这个服务的记录文件就直接给他放置在 /var/log/messages 里面啦，所以来看看里面的几行吧！

```
[root@test root]# tail -n 15 /var/log/messages | grep named
Dec  5 02:33:33 test named[3010]: starting BIND 9.2.1 -u named
Dec  5 02:33:33 test named[3010]: using 1 CPU
Dec  5 02:33:33 test named[3015]: loading configuration from
'/etc/named.conf'
Dec  5 02:33:33 test named[3015]: no IPv6 interfaces found
Dec  5 02:33:33 test named[3015]: listening on IPv4 interface lo,
127.0.0.1#53
Dec  5 02:33:33 test named[3015]: listening on IPv4 interface eth0,
192.168.1.2#53
Dec  5 02:33:33 test named[3015]: running
Dec  5 02:33:33 test named: named startup succeeded
```

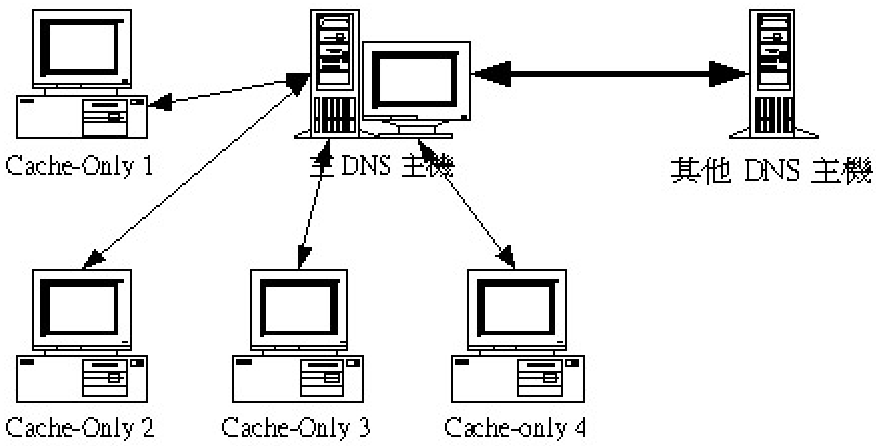
7. 呵呵！看起来似乎是没有问题的样子了！好了！那么就直接来测试看看吧！

8. ---

9. 测试：
   这部分请参考： Client 端的测试项目

特别说明：Forwarders 的好处与问题分析
关于 forwarder 的好处与坏处，其实有很多种的意见！大致的意见可分为这两派：

- 利用 Forwarder 的功能来增进效能的理论：
  这些朋友们认为，当很多的下层 DNS 主机都使用 forwarder 时， 那么那个被设定为 forwarder 的主机，由于会记录很多的信息记录(请参考图三的说明)， 因此，对于那些下层的 DNS 主机而言，会增快很多，亦即会节省很多的查询时间！基本上，这些基本的流程可以看成如下图所示：



图六、Forwarder 参数的运作说明

  所有的 cache-only 都设定 forwarder 为『主 DNS 主机』那一部，则由于主 DNS 主机已经记录了较多的信息了(每个人都来要求嘛！)所以，当其它人来要求相同的查询数据时，则主 DNS 那部机器将会直接由其 cache 当中读取，因此，查询效率就变快了！

- 利用 Forwarder 反而会使整体的效能降低：
  但是另外一派则持相反的见解！这是因为当主 DNS 本身的『业务量』就很繁忙的时候，那么您的 cache-only 主机还向他要求数据，那么因为他的数据传输量太大，频宽方面可能负荷不量，而太多的下层 DNS 又向他要求数据，所以他的查询速度会变慢！因为查询速度变慢了，而您的 cache-only 主机又是向他提出要求的，所以自然两边的查询速度就会同步下降！

很多种说法啦！鸟哥本人也觉得很有趣哩！只是不知道哪一派较正确就是了 >_<"" ，不过可以知道的是，如果上层的 DNS 速度很快的话，那么他被设定为 forwarder 时，或许真的可以增加不少效能哩！

DNS 主机的详细设定：

接下来我们就来架设一部完整的 DNS 主机吧！如同前面说的，我们必须要设定的档案有几个呢？

1. /etc/named.conf
2. /var/named/named.root
3. /var/named/named.localhost
4. /var/named/named.127.0.0
5. /var/named/named.正解档案
6. /var/named/named.反解档案

大概就是这几个！要注意的是，除了第一个 /etc/named.conf 的档名是预设的之外，其它的档名都是在 /etc/named.conf 里面设定的！那么底下我们就以鸟哥家里的 DNS 主机设定来说明一下俺是如何设定我的 domain name 啰！要注意的是，这里的 DNS 设定是『私有网域的设定』状态， 如果您刚刚看过了『授权』的概念，那么将会知道， 底下我所设定的皆是属于『不合法的 DNS 主机』，这意味着我的 DNS 主机只能够向外查询，但是别人是查不到我的 DNS 主机里面的设定内容的！除非他使用我的 DNS 主机的 IP 啰！不过，嘿嘿！我使用的是私有 IP ，想要使用我的 DNS ！哈哈！门都没有～

---

1. 手动规划 hostname 与 IP 的对应表：
   在作任何事之前，先动手设计一下是好事呐！我假设我的 domain name 是 vbird.tw 而网域为 192.168.1.0/24，主机的名称配合 domain name 来设计的共有三部计算机，分别为：

计算机系统	计算机 IP	计算机名称	说明
MDK 10.0	192.168.1.2	mdk.vbird.tw forum.vbird.tw www.vbird.tw ftp.vbird.tw	这部计算机是主要的 DNS 主机，我的主要名称是 mdk.vbird.tw ，其它三个则是『主机别名！』
Win2K	192.168.1.100	win2k.vbird.tw	这部主机名称是记录在 mdk.vbird.tw 里面的纪录数据。
WinXP	192.168.1.200	winxp.vbird.tw	这部主机名称是记录在 mdk.vbird.tw 里面的喔！

2.
   要注意的是，在 mdk.vbird.tw 那部机器中，因为该计算机的用途相当的多， 所以我希望那一部主机有多个名称！那么因为目前又只有一个正解的领域，所以就仅设定了这个 domain 里面的三个别名了！所以那部主机总共有四个名字呐！
3.
   ---
4. 设定简易的 /etc/named.conf 档案：
   还记得上面提过的，这个档案才是主要的设定档，而其它的 hostname <--> IP 则是在各个 zone 的设定档中！那么这个档案主要的设定首先在于针对主机的设定， 这一点刚刚 forwarder DNS 主机已经说过了！那另一个则是在定义每一个 zone 的文件名称与该设定 domain 的 『 Type (类型)』，底下介绍三种主要类型， 分别为 master (主要设定档) 、 hint (就是 root 啦) 以及 slave (针对 master 来进行数据同步化的设定文件)。 好了，那么如果以我们刚刚上面规划的设定来看，那么应该有的 /etc/named.conf 设定就会变成底下的模样了：

```
[root@test root]# vi /etc/named.conf
// 设定整体的主机规划！重点在 directory 的意义！
options {
        directory "/var/named";
//这个是在规定『我的正反解档案放置的目录』
        forwarders {
                168.95.1.1;
//不管怎么说，俺就是喜欢 forwarder 的设定
        };
        pid-file "/var/run/named/named.pid"; //每个版本可能都不同！
        allow-query { any; };    //是否允许他人查询？当然啦！
        allow-transfer { none; };
};    //上面这个设定项目，主要针对 Master/Slave 的架构，
      //亦即是否允许来自 slave 端的要求而提供整个 zone 的传送！
      //近年来由于一些安全性的问题，所以，这里我们建议将他设定为 none

// 首先定义出 . (root) 这个 hint type 的档案内容！
zone "." {                   //看到了没！？这个就是所谓的 zone 啦！
        type hint;           //选择的 type 为 hint (root . 专用)
        file "named.root";   //设定档案的档名！预设为 named.root
};                           //有的时候也可能是 named.ca 喔！

// 再来则是定义出 localhost 的正反解了！很简单啦！就是 127.0.0.1 而已
zone "localhost" {              //这个 zone 表示设定档的预设 domain
           //name 为 localhost 的意思喔！这里请『特别』搞清楚！
        type master;            //主要的在本机的设定档！
        file "named.localhost"; //檔名！可以随自己高兴随便取！
};
zone "0.0.127.in-addr.arpa" {  //反解的 IP 网段！那个 in-addr.arpa 是
                               //固定的 IP 段写法！
        type master;
        file "named.127.0.0";
};

// 定义出我自己的这一组正反解设定！
zone "vbird.tw" {            //我的 zone 的 domain name 为 vbird.tw
        type master;
        file "named.vbird.tw";
};
zone "1.168.192.in-addr.arpa" {
        type master;
        file "named.192.168.1";
};
```

要特别留意的是：

- options
  - directory：这个设定值就是在规范每一个 zone 的档案放置的目录。 举例来说，如果是 localhost 的正解檔，亦即是 named.localhost 时， 那么这个档案的放置位置就是在 /var/named/named.localhost 啦！这样可以了解吗？ 未来您可以自己改变自己档案放置的地方，就可以分的比较清楚！不过，习惯上还是放置在 /var/named 里面！
  - pid-file：指的是每一个 services 的记录自己的 PID (Process ID) 的档案啰！这个档案通常用在重新启动或者是 reload 整个 services 最常被使用到的！因为可以使用 kill -1 PID 来重新启动啊！嗄！忘记什么是 PID ？赶快拿出『鸟哥的 Linux 私房菜 -- 基础学习篇』 复习一下！
  - forwarders (不要忘记那个 s 喔！)： 就是要设定往前寻找的那个『合法』的 DNS 啰！每一个 forward 的主机之 IP 都需要有『 ; 』来做为结尾！
- 关于 . (root) 的内容：
  root 最重要的就是那个 hint 的 type 啦！记得写对喔！
- 关于 localhost 的正反解：
  正反解的名称都可以随意设定，不过，要特别留意的就是那个 zone 后面接的其实就是『 domain name 』！这个 domain name 未来在 zone 的设定档当中会使用得很频繁喔！
- 关于其它 domain 的正反解：
  其实与 localhost 没有什么不同的，就只是不同的 domain name 就是了！
- 反解的写法：
  反解的 Zone 的写法较为特殊，他必须要将 IP 反过来写的，例如 127.0.0.0/24 这个 C class 的网域，要写的话，则必须要反过来写成 0.0.127 这样的形式！其中需要注意的是，最后面务必要加上『 in-addr.arpa』的咚咚！不要忘记了喔！

---

5. 设定 . (root) 的内容：
一般来说，如同图三的流程解说一般，如果在本机上查询不到某部主机的 IP， 而我们的 DNS 主机又没有设定 forwarders 时，那么 DNS 主机通常就是直接到 . (root) 去查询啰！但是我们怎么知道 root 在哪里呢？总还是需要有 IP 吧！没错啰！ 这时候就需要 hint 这个 type 来支持啦！一般来说，在 BIND 这个套件释出时，都会附上 . 也就是 named.root (或 named.ca) 这个档案的，如果没有的话，没有关系，我们可以连接上管理国际 domain name 的机器，那就是 rs.internic.net 这部机器去下载啰！您可以这样做：

```
[root@test root]# ftp rs.internic.net
Connected to rs.internic.net.
Name (rs.internic.net:root): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: <==your password
```

```
230 User ftp logged in.  Access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd domain
ftp> get named.root
ftp> bye
```

赶紧来看一下这个 named.root 档案的内容吧！

```
[root@test root]# vi named.root
; 抱歉，版权宣告部分先省略～
;
; formerly NS.INTERNIC.NET
;
.                          3600000   IN   NS     A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.        3600000        A      198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                          3600000        NS     B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.        3600000        A      128.9.0.107
;
; formerly C.PSI.NET
;
; 以下省略
```

共有若干个的主机，注意喔！那个『 ; 』是设定档的批注，与 /etc/named.conf 又不一样！特别留意，不要搞错了！然后您会发现每个『 . 』都有个 NS 与 A 的对应， 注意看到粗体字的那两行，第一行意思是 . 的 name server (NS) 为 『A.ROOT-SERVERS.NET. 』最后面有没有加上 . 是不一样的！特别留意！而这个 nameserver 的 IP (用 A 对应) 为 198.41.0.4 ！！其它的用途我们在下一个档案再来谈！总而言之， 这个档案不要去改他 ！因为这个是国际上通用的资料，不能修改啦！改了反而会找不到 . 哩！

---

6. 设定本机端 (localhost) 的正解档案：
   每一部机器都有 localhost 嘛！所以呢，我们就先来针对 localhost 这个网域的计算机对应来设定一下啰！ 而由于 localhost 通常其对应的 IP 就是 127.0.0.1 ，所以就直接给他正解有这两个就行啦！ 但是要特别留意的是，『我的正解要找哪一个 nameserver ！？』因此，在底下的设定档当中， 最重要的其实就是要使用 NS 这个 nameserver 的『主机名称』标志啦！

```
[root@test root]# cd /var/named
```

```
[root@test named]# vi named.localhost
$TTL    600                 ; 这个跟清除 cache 的时间有关系！单位是秒！
@       IN      SOA     localhost.      root.localhost. (
                2002120601  ; Serial  与 master 及 slave 是否同步有关！
                        ; 一般而言，如果这个数值变大了， slave 才会同步更新！
                28800       ; Refresh 定义出 slave 多久会主动的检查 serial
                        ; 的值，以便主动的更新数据库！
                14400   ; Retry   定义出，如果 slave 没有连上 master DNS
                        ; 主机则多久之后会重新再次的主动检查！
                720000      ; Expire  如果一直没有连接上 mater ，那么到了
                        ; 这个时候 slave 就会放弃检查的动作了，不再更新！
                86400 )     ; Minimum 这个其实就是 TTL 啦！如果您没有定义
                        ; TTL ，那么 TTL 的值就以这个来设定！
; 开始设定正解的信息内容：
@               IN      NS      localhost. ; 特别留意最后面有个 . 喔！
localhost.      IN      A       127.0.0.1
; A 是正解里面 hostname 对应 IP 的标志
```

上面有很多的怪怪的字眼，我们得要先说明一下，否则后面您会『雾煞煞』的！

符号	说明
$TTL	1. 这个东西主要在：『定义出向外查询的数据可以记录在 DNS 的 cache 当中多久』的意思； 2. 后面接的数字单位为秒； 3. 通常这个数字如果定义太大的话，例如一天(86400)时，那么当别人更改了他的 DNS 讯息时， 由于您的 cache 更新时间为一天，所以得要一天之后 cache 当中的数据才会被取代，因此，在一天之内， 您查询到的信息『都会是旧的！』 4. 但是这个数字如果定的太小的话，例如五分钟(300)那么这部 DNS 将会不断的向外要求数据， 则负荷会变的较大啦！ 5. 其实，除非是在测试阶段，不然的话，通常都会建议定义一天的 cache 时间啰！ 6. 注意：某些套件上面并不能定义这个咚咚！
@	这个就是 zone 定义出的那个咚咚啦！以这个档案内容为例，因为我们在 /etc/named.conf 当中就是定义出 localhost 这个 domain name 为一个 zone 的，因此，呵呵！在这里， 这个符号就代表 localhost 啦！
SOA	7. 这个是 Start of Authority 开始设定的内容的意思啦！也就是接在后面的设定要开始了！ 请注意，这个咚咚在每个『zone 的设定档』当中都会存在！所以，每个 zone 的设定都一样即可！

	8. 在 SOA 后面会接两个咚咚，第一个为主机名称(localhost.)，请特别留意那个 localhost 后面有个小数点 (.) 这个东西很重要！他代表『一个完整的 hostname + domain name 了』！ 如果没有加上 (.) 的话，那么就表示该文字『仅为 hostname ，还需要加上 domain name 』！ 这里是新手最容易出现的错误喔！第二个为管理员的 e-mail ！因为不能使用 @ (已经是特殊符号了)， 所以这里也同样的以 (.) 来取代！例如上面我以 root@localhost 来做为我的 e-mail ，所以就写成了 root.localhost. ，同样的，最后面有个 (.) 喔！ 9. 在最后，会有小刮号 ( ) 括起了五个数字，这五个数字除了最后一个与 TTL 有关之外， 其它的都跟 slave 与 master 的资料同步运作有关！ ▪ Serial ：这个数字仅是用来做为 master 与 slave 之间的 update 的参考数值也就是说，当 Slave 的 serial 小于 Master 时， 那么 update 才会动作！由于担心设定者的设定技巧问题，因此通常我们以时间来做为 Serial 的订定依据，例如 2002 年 12 月 6 日第一次设定，可以写成 『2002120601』请注意，这个数字不可超过 10 个数字。 ▪ Refresh ：命令 slave 多久进行主动更新的时间； ▪ Retry ：如果到了 Refresh 的时间，但是 slave 却无法连接到 master 时， 那么在多久之后，slave 会再次的主动尝试与主机联机； ▪ Expire ：如果 slave 一直无法与 master 连接上，那么经过多久的时间之后， 则命令 slave 不要再连接 master 了！ ▪ Minimun ：这个就有点像是 TTL 啦！ 另外，各个值是有大小限制的，他们的限制是： ▪ Serial <= 2^32 ▪ Refresh >= Retry * 2 ▪ Refresh + Retry < Expire ▪ Expire >= Retry * 10 ▪ Expire >= 7Days
NS	10. 表示 name server 的意思，后面接的都是『hostname 或 FQDN』这个表示前面的 domain 是由后面的这个主机所管理的啦！ 11. 『 @ IN NS localhost. 』这一行的意思是说，@ (zone ，亦即是 localhost 这个 domain) 的管理的 Name Server 为 localhost 这部主机，请注意，那个 localhost 后面一定要接 (.) 才行！为什么呢？因为如果没有加上 (.) 的话，那么主机名称将会变成 localhost.localhost ！ Why ？ 这是因为

	BIND 预设情况中，没有写 . 的话，那么则表示该名称为 Hostname 而已，需要再加上 domain name 才行！ 12. 由于 Name Server 为主机的名称，所以后续还要加上这个 name server 的正解的 IP 对应(就是底下要谈的 A)才行！
A	这是正解的符号啦！也就是说，前面的 localhost. (还是得要注意那个 .)所对应的 IP 为 127.0.0.1 的意思啦！
.	呵呵！再次的给他强调下，在 BIND 的设定档当中，关于主机名称的话，最后面有没有加上 . 是差很多的！加上了 . 表示这个『完整的主机名称，亦即是 hostname + domain name 』了， 如果没有加上 . 的话，表示该名称仅为『 hostname 』而已！切记切记！

这样可以了解上面的意思了吗？呵呵!这个是最基础的几个项目喔!千万要记得!不要忘记了～ 好了，知道了正解之后，我们要来谈一谈那个反解的东西啦！

---

7. 设定本机端 (localhost) 的反解档案：
反解跟正解一样，还都需要 SOA 的标号，也需要 NS 这个咚咚，唯一不同的大概就是由 IP 对应成为 hostname 的不同了吧！

```
[root@test root]# cd /var/named
[root@test named]# vi named.127.0.0
# 这个文件名称是在 /etc/named.conf 里面设定
$TTL    600
; This is about DNS server's settings
@       IN      SOA     localhost.      root.localhost. (
                2002120601      ; Serial
                28800           ; Refresh
                14400           ; Retry
                720000          ; Expire
                86400 )         ; Minimum
; The server's infomations
@               IN      NS      localhost.
1               IN      PTR     localhost.
```

上面前几行跟前面一样，就不提了！只有最后一行不一样，那个是什么呢？

符号	说明

PTR	0. 这是反解的符号啦！主要还是在于 IP 对应主机名称的咚咚！要注意的是，由于这个档案的 zone 为 127.0.0 ，所以我们只要加一个数字(最后一个数字)就可以啦！而那个 1 表示的就成为了 127.0.0.1 啰！ 1. 那么万一今天我们规划的是 B Class 的 zone 呢？例如 127.0 这样的 zone 呢？很简单啦！ 就填两个数字即可！也就是 0.1 啰！ 2. 最重要的东西就是：在规划 zone 的时候，是很重要的，而反解的 zone 的名称最后需要接上 in-addr.arpa，这点也请千万不要忘记了！

因为这个 domain(localhost) 就只有一部机器，所以我们很简单的就可以将他设定完成了！ 底下，我们将要设定我们自行假设定三部主机喔！

---

8. 设定 domain name 的正解：
   再来设定的就是刚刚我们先前提到的三部主机了，您可以这样设定：

```
[root@test root]# cd /var/named
[root@test named]# vi named.vbird.tw
# 这个文件名称是在 /etc/named.conf 里面设定
$TTL    600
; 跟上面提到的一样，设定主机的一些基本信息！
@       IN      SOA     mdk.vbird.tw.        root.mdk.vbird.tw.    (
                        2004102901      ; Serial
                        28800           ; Refresh
                        14400           ; Retry
                        720000          ; Expire
                        86400    )      ; minimum
; 主机的设定参数部分
@               IN      NS      mdk.vbird.tw.
; 这个 zone (vbird.tw) 的主机
@               IN      MX 10   mdk.vbird.tw.
; 邮件转递的主要邮件主机！
mdk             IN      A       192.168.1.2
mdk             IN      TXT     "The testing DNS server"
; 仅是说明文件
phorum          IN      CNAME   mdk
;注意这行与下一行的写法！
www             IN      CNAME   mdk.vbird.tw.
; 其它主机的设定信息上面
```

```
win2k           IN      A       192.168.1.100
win2k           IN      HINFO   "AMD-K6-III""Windows 2000"
winxp           IN      A       192.168.1.200
```

除了先前的 master 与 slave 相关的时间参数之外，还有那个重要的 NS 以及 A 这些参数之外，里面的参数主要有底下几个新鲜玩意儿：

符号	说明
SOA	特别注意到 SOA 那一行的设定喔！因为我们要设定的已经是 vbird.tw 这个 Zone 了，所以请修改一下您的主机名称，还有 DNS 主机的管理员邮件地址喔！ o 另外，还是再次的提醒那个 . 是什么东西！
MX	o 这个东西就是 Mail eXchanger (MX) 的简写，他的用途在使用于邮件主机时， 需要的信件转递站！用于一般主机是没有多大的影响，但是对于 mail server 则有相当重要的影响哩！ o 如果不知道如何使用这个玩意儿，没有关系，您可以直接将您的主机名称(FQDN)写入！ 以我上面的例子来说，我就将邮件主机写成我自己的主机，注意，最好是 FQDN 喔！ o 如果您的邮件主机没有 MX 这个设定其实也没有关系啦，信件还是可以传送到达的，但是， 有时后就是会比较慢一些些收到对方寄来的信件就是了！ o 请注意 MX 后面要接上一个数值喔！您可以设定多个邮件主机，但是请特别留意的是， 被设定的邮件主机必须要能够支持您的邮件之 relay 才行，否则设定会变成无效的！
TXT	o 这个东西在进行『说明』而已！亦即是前面那部主机的一些信息。 o 特别注意的是，没事的话，『信息不要写得太详细，有的时候甚至应该要写些错误的讯息！』 为什么呢？如果写得太详细的话，那么那些个 cracker 不就很简单的就可以将您的网站信息取得， 并进而入侵了吗？ @_@
CNAME	这个东西就是设定主机别名的咚咚啦！因为我们的主机有很多个名字，没有必要为每个名字都建立一个 A 的标号，这个时候，我们就可以使用 CNAME 来设定另外一个别名！以上面为例，我设定了两个别名在我的主机上面，特别留意的是两个 CNAME 的写法都指向同一部机器，上面关于 CNAME 的那两行最大差异性在于写的是否为 FQDN 与后面有没有加上 .

	这个标志啦！ ^_^
HINFO	这个东西后面接两个咚咚，第一个接的是硬件的等级，第二个接的则是操作系统， 这两个咚咚最好不要用在公开的 DNS 主机上面，跟 TXT 一样的问题啦！如果要设定的话， 最好使用双引号分隔开来喔！

这样应该就设定妥当啰！请额外注意喔！在 DNS 的正解部分，他的重要信息特别的多， 比较难设定的意思就对了～所以，您需要特别留意每个设定值是否为正确喔！一般而言， 我们会建议大家，设定完成之后，并且执行完启动的 script ，千万要记得去 /var/log/messages 里头看一看有没有错误讯息喔！

---

9. 设定 domain name 的反解：
   设定反解要简单的多了～只要找到对应的 hostname 即可：

```
[root@test root]# cd /var/named
[root@test named]# vi named.192.168.1
# 这个文件名称是在 /etc/named.conf 里面设定
$TTL    600
@       IN      SOA     mdk.vbird.tw.        root.mdk.vbird.tw.   (
                        2004102901      ; Serial
                        28800           ; Refresh
                        14400           ; Retry
                        720000          ; Expire
                        86400   )       ; minimum
; 其它主机的信息！
@       IN      NS      mdk.vbird.tw.
2       IN      PTR     mdk.vbird.tw.
; The following is about other hosts
100     IN      PTR     win2k.vbird.tw.
200     IN      PTR     winxp.vbird.tw.
```

很简单吧！就是 IP 的对应即可！

---

10. 启动 named 与 port 及讯息确认：
    又到了启动的时刻了！加油！

```
1. 开始启动！
[root@test root]# /etc/rc.d/init.d/named start
Starting named:                  [  OK  ]
```

```
2. 关于 port 观察：
[root@test root]# netstat -tuln | grep 53
tcp        0      0 192.168.1.2:53          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
udp        0      0 192.168.1.2:53          0.0.0.0:*
udp        0      0 127.0.0.1:53            0.0.0.0:*

3. 关于讯息的内容！
[root@test root]# tail -n 15 /var/log/messages
Oct 29 17:30:33 test named[27159]: using 1 CPU
Oct 29 17:30:33 test named[27159]: loading configuration from
'/etc/named.conf'
Oct 29 17:30:33 test named[27159]: listening on IPv4 interface
lo, 127.0.0.1#53
Oct 29 17:30:33 test named[27159]: listening on IPv4 interface
eth0, 192.168.1.2#53
Oct 29 17:30:33 test named[27159]: zone 0.0.127.in-addr.arpa/IN:
loaded serial 2002120601
Oct 29 17:30:33 test named[27159]: zone 1.168.192.in-addr.arpa/IN:
loaded serial 2004102901
Oct 29 17:30:33 test named[27159]: zone localhost/IN: loaded serial
2002120601
Oct 29 17:30:33 test named[27159]: zone vbird.tw/IN: loaded serial
2004102901
Oct 29 17:30:33 test named[27159]: running
Oct 29 17:30:33 test named[27159]: zone vbird.tw/IN: sending notifies
(serial 2004102901)
Oct 29 17:30:33 test named: named start succeeded
```

---

11. 可能的设定错误问题：
基本上，DNS 算是很难设定的一个 Server 了，所以在除错方面请务必小心！ 他的设定错误通常有两种情况：

    - 语法设定错误：
    这个问题比较好解决，因为在 /var/log/messages 里面都已经说的很清楚了！按照内容去修订即可；
    - 逻辑设定错误：
    这个就比较困扰了！为什么呢？因为他主要发生在您设定 DNS 主机的时候，考虑不周所产生的问题！例如忘记加上 (.) ， 系统不会显示错误讯息，但是却会造成查询的误判，而 MX 设定的主机名称错误，也不会出现有问题的讯息，但是 mail server 就是会收不到信等等～这些错误都需要很详细的 DNS client 的测试才能知道问题的所在。

我们这里先就语法设定错误方面进行介绍，至于逻辑设定的问题，那个就需要多多的进行测试才能知道了～
底下的错误讯息都会记录在 /var/log/messages 里面喔！

```
Dec 10 11:34:21 test named[31185]: /etc/named.conf:18: missing ';'
before '}'
Dec 10 11:34:21 test named[31185]: loading configuration: failure
Dec 10 11:34:21 test named[31185]: exiting (due to fatal error)
# 这样的错误就是发生在 /etc/named.conf 的地 18 行，忘记加上 ;
# 符号了！去修正即可！

Dec 10 11:37:20 test named[31236]: dns_rdata_fromtext:
named.localhost:9: near eol: unexpected end of input
Dec 10 11:37:20 test named[31236]: zone localhost/IN:
loading master file named.localhost: unexpected end of input
# 这样的错误通常就是发生在 named.localhost 的第 9 行处，去修正看看，
# 通常在前面几行很有可能是 SOA 后面接的五个数字没有写全的原因！

Dec 10 11:42:28 test named[31338]: dns_master_load:
named.localhost:14: unexpected end of line
Dec 10 11:42:28 test named[31338]: dns_master_load:
named.localhost:13: unexpected end of input
Dec 10 11:42:28 test named[31338]: zone localhost/IN:
loading master file named.localhost: unexpected end of input
# 同样的，告诉您在该档案 named.localhost 有问题！请查证！
```

通常最大的原因真的就是打错字啦！总之，赶紧去看看 /var/log/messages 的内容，一定可以让您更了解错误的问题喔！

---

12. 测试：
这样就设定完成了！接下来，请查阅 在 Client 端的测试 吧！

Master/Slave 架构的详细设定：

如同我们在前面提到的，如果您有多部 DNS 主机管理同一个领域名称的话， 那么为了节省维护成本，以 Master/Slave 架构来规划主机的配置实在是不错的一个方式！ 因为，如果需要修改 zone 的主机名称与 IP 的对应数据，则只要更改 Master 那部主机即可！

在 Master/Slave 的架构下，Master 的设定与前一小节的设定是相同的， 只不过需要设定预定被传送的

zone 的 allow-transfer 项目值而已！ 至于 slave 部分，他不需要预定要传送的 zone 的档案，因为该档案会自动产生～ 底下我们就来设定一下吧！

---

1. 草案规划
   同样的以上一小节的案例来说明，我的 Master 管理的是 vbird.tw 这个领域名称， 他所需要设定的档案有六个，分别是设定档 /etc/named.conf，以及 zone 的数据表， 在 /var/named 里面的 named.root， named.localhost， named.127.0.0， named.vbird.tw named.192.168.1 等等。

   至于 slave 方面，同样需要设定 /etc/named.conf ，不过，在 zone file 方面，他仅需要 named.root， named.localhost， named.127.0.0 而已，另外两个档案会自动建立！

---

2. Master 的设定
   Master 的设定与上一小节几乎完全相同，只是要变一个设定值，请修改 /etc/named.conf 成为这样：

```
....(略)....
zone "vbird.tw" {
        type master;
        file "named.vbird.tw";
        allow-transfer { 192.168.1.21; };
};
zone "1.168.192.in-addr.arpa" {
        type master;
        file "named.192.168.1";
        allow-transfer { 192.168.1.21; };
};
....(略)....
```

   亦即我仅允许 192.186.1.21 取得我的 zone file 的所有内容之传送啊！其它的几个档案都与前一小节相同。

---

3. Slave 的设定
   至于 Slave 的设定方面，在 /var/named 里面的 named.root，named.localhost，named.127.0.0 都可以直接由 Master 复制过来，而 /etc/named.conf 也可以直接复制过来， 只要修改成底下这样即可：

```
[root@test2 root]# vi /etc/named.conf
....(略)....
```

```
zone "vbird.tw" {
        type slave;
        file "named.vbird.tw";
        masters { 192.168.1.2; };
};
zone "1.168.192.in-addr.arpa" {
        type slave;
        file "named.192.168.1";
        masters { 192.168.1.2; };
};
....(略)....
```

看到了吗？在 Master 的部分允许来自 192.168.1.21 这个 slave 的要求，而 slave 就是向 192.168.1.2 这个 master 要求 zone file 的传送！在这样的设定完毕之后， 两边均同时启动 named ，在 Slave 就会自动的建立两个 zone file 啰！

未来，您要增加其它的主机名称与 IP 对应的数据，只要在 Master 那部主机上设定好， 并重新启动 named ，那么 Master 会依据 serial number 来判断是否通知 slave 前来更新， 此外， Slave 也会依据设定的时间值，自动的来 Master 读取数据喔！ 如此一来，管理上面是否真的比较容易呢！ ^_^y

---

Client 端的设定：

说完了在 DNS Server 端的设定，接下来，我们再来聊一聊关于 Client 端的设定与测试！ 从前面的说明里面，我们晓得主机名称对应到 IP 有两种方法，早期的方法是直接写在档案里面来对应， 后来比较新的方法则是透过 DNS 架构！那么这两种方法目前的使用状态是怎样的呢？

- 档案设定：
  既然已经完成了 DNS 主机的设定，接下来自然要进行 Client 端的联机测试啦！要怎么测试呢？底下有几个档案请特别留意喔：

  - /etc/hosts ：刚刚上面就提过了，这个是最早的 hostname 对应 IP 的档案；
  - /etc/resolv.conf ：这个重要！就是 DNS 主机的 IP，您的 Client 就是利用这里面设定的 IP 去追踪名称解析的。
  - /etc/nsswitch.conf：这个档案则是在『决定』先要使用 /etc/hosts 还是 /etc/resolv.conf 的设定！

一般而言， Linux 的预设 hostname 搜寻都是先 /etc/hosts 来的， 为什么呢？您可以查看一下 /etc/nsswitch.conf ，并找到 hosts 的项目：

```
[root@test root]# vi /etc/nsswitch.conf
```

```
hosts:      files nisplus nis dns
```

上面那个 files 就是使用 /etc/hosts 而最后的 dns 则是使用 /etc/resolv.conf 的 DNS 主机 IP 搜寻啦！因此，您可以先以 /etc/hosts 来设定 IP 对应ㄋㄟ！ 当然啦，您也可以将他调换过来，不过，总是 /etc/hosts 比较简单，所以将他摆在前面比较好啦！

好啦，既然我们是要进行 DNS 测试的，那么 /etc/resolv.conf 的内容，自然就要填写我们自己的 IP 啰！所以您应该这样写：

```
[root@test root]# vi /etc/resolv.conf
nameserver 192.168.1.2
nameserver 168.95.1.1
nameserver 139.175.10.20
```

DNS 主机的 IP 可以设定多个，这可以让您的个人计算机有备援的功能！举例来说， 我上面共设定了三部主机作为我的 DNS 查询，当 192.168.1.2 那部主机挂点时， 我的 Client 计算机会立刻以第二部主机作为 DNS 查询的主要主机。所以， 通常我们都会建议人家在这个档案内可以设定三个左右的 DNS 主机名称！以保不时之需啊～

另外，上面三个 DNS 的 IP 那个会先被使用？当然是照顺序来的～ 所以会先以 192.168.1.2 那部主机来查询，若 192.168.1.2 挂了，才会使用 168.95.1.1 那部来查询。

有个观念得要提醒一下，我们常常建议人家，在自家设的， 没有经过合法授权的 DNS 最好不要以 Internet 上面已经存在的领域名称来练习架设！ 举例来说，假设今天我以我的 192.168.1.2 那部机器来架设 *.yahoo.com 的领域， 也就是说，在 192.168.1.2 那部机器是有提供 yahoo.com 的 zone 的资料的(注：那是虚拟的～) 但是因为我将 192.168.1.2 放置在第一位，导致每次的查询其实 yahoo.com 这个领域的数据都是直接由 192.168.1.2 所提供，这很不好～

因为可能会造成您的客户端的不便～

好了，我们要测试我们的 DNS 主机设定是否正确啰！

- 测试 DNS 设定：
  测试 DNS 的程序有很多，我们先来使用最简单的 host 吧！然后还有 nslookup 及 dig 哩！

host

```
语法：
[root@test root]# host [-a] [FQDN] [server]
[root@test root]# host -l [domain] [server]
参数说明：
-a ：所有的信息都列出来，列出的信息包含有 TTL
     DNS 主机的 IP，所欲寻找的主机的 IP 等等，数据丰富啊！
-l ：将后面接的 domain 内的所有的 host 都列出来！不过，这个项目要能够
     被利用，就必须要有 allow-transfer 的项目在 /etc/named.conf 里面被启动！
server：这个参数可有可无，当想要利用非 /etc/resolv.conf 内的 DNS 主机
        来查询主机名称与 IP 的对应时，就可以利用这个参数了！
范例：

# 强制以我的 192.168.1.2 这部 DNS 主机来查询
[root@test root]$ host mdk.vbird.tw 192.168.1.2
Using domain server:     <==如果加上最后的 192.168.1.2 这个 server 的参数
Name: 192.168.1.2        <==那就会出现这三行字眼，表示用的 DNS 主机不是
Address: 192.168.1.2#53  <==/etc/resolv.conf 所欲设的 DNS 主机！

mdk.vbird.tw has address 192.168.1.2
# 很简单吧！立刻找到 IP 啰！

[root@test root]$ host -a mdk.vbird.tw 192.168.1.2
Trying "mdk.vbird.tw"
Using domain server:
Name: 192.168.1.2
Address: 192.168.1.2#53
Aliases:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41087
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1,
ADDITIONAL: 0

;; QUESTION SECTION:
;mdk.vbird.tw.                  IN      ANY
```

```
;; ANSWER SECTION:
mdk.vbird.tw.           600     IN      A    192.168.1.2
mdk.vbird.tw.           600     IN      TXT "The testing DNS"

;; AUTHORITY SECTION:
vbird.tw.               600     IN      NS   mdk.vbird.tw.

Received 95 bytes from 192.168.1.2#53 in 6 ms
# 在这个范例当中，我们可以看到整个显示出的讯息包括有几个部分：
# HEADER(标题)部分==>显示查询的内容有哪些，包括一个 query, 两个 answer
#                    及一个验证部分。
# QUESTION(问题)====>显示所要查询的内容，因为我们是查询 mdk.vbird.tw
#                    所以这里自然就是显示这个讯息。
# ANSWER(回应)======>依据刚刚的 QUESTION 去查询所得到的结果，因为在我们的
#                    设定当中仅有设定了 A 与 TXT 的标签，所以这里自然就....
# AUTHORITY(验证)===>由这里我们可以查阅 vbird.tw 这个领域是由 mdk.vbird.tw
#                    来设定的～
# 里面那个 600 是什么呢？很简单，他就是我们所设定的 ttl 那个数值啦！

[root@test root]$ host -l vbird.tw 192.168.1.2
Using domain server:
Name: 192.168.1.2
Address: 192.168.1.2#53
Aliases:

vbird.tw name server mdk.vbird.tw.
mdk.vbird.tw has address 192.168.1.2
win2k.vbird.tw has address 192.168.1.100
winxp.vbird.tw has address 192.168.1.200
# 上面的信息可就熟悉多了吧？！没错！那就是我们在
# /var/named/named.vbird.tw 里面的设定值啊！
# 不过，并不是所有的 domain 都可以作这样的事情～举例来说，如果我们下达：
# host -l tw.yahoo.com 则将会得到：
# Host tw.yahoo.com not found: 5(REFUSED)
# ; Transfer failed.
# 这样的响应，这是因为在 /etc/named.conf 里面并没有设定 allow-transfer
# 那个设定选项的原因啊！
```

事实上，使用 host 几乎就可以达到我们的要求了～也不需要什么其它的指令～ 不过，其实还是得要知道其它的查询指令啦～

nslookup

```
语法：
[root @test root]# nslookup [FQDN]
[root @test root]# nslookup
参数说明：
如果在 nslookup 后面没有加上任何主机名称或 IP ，那将进入 nslookup 的查询功能
在 nslookup 的查询功能当中，可以输入两个参数来进行特殊查询：
set type=any ：列出所有的信息『正解方面设定档』
set type=mx  ：列出与 mx 相关的信息！
范例：

[root@test named]# nslookup win2k.vbird.tw 192.168.1.2
Server:         192.168.1.2
Address:        192.168.1.2#53

Name:   win2k.vbird.tw
Address: 192.168.1.100
# 单纯的将 hostname 与 IP 对应列出而已，不过，
# 还是会将查询的 DNS 主机的 IP 列出来的！

[root@test named]# nslookup  <==进入 nslookup 查询画面
> 192.168.1.2<==执行反解的查询
Server:         192.168.1.2
Address:        192.168.1.2#53

2.1.168.192.in-addr.arpa        name = mdk.vbird.tw.

> www.vbird.tw   <==执行正解的查询
Server:         192.168.1.2
Address:        192.168.1.2#53

www.vbird.tw    canonical name = mdk.vbird.tw.
Name:   mdk.vbird.tw
Address: 192.168.1.2
> tw.yahoo.com   <==执行非本机上的查询
Server:         192.168.1.2
Address:        192.168.1.2#53

Non-authoritative answer:
tw.yahoo.com    canonical name = vip1.tw.tpe.yahoo.com.
Name:   vip1.tw.tpe.yahoo.com
Address: 202.43.195.52
> set type=any   <==显示所有查询的信息
> mdk.vbird.tw
```

```
Server:         192.168.1.2
Address:        192.168.1.2#53

Name:   mdk.vbird.tw
Address: 192.168.1.2
mdk.vbird.tw    text = "The testing DNS server"
> exit
```

在上面的案例当中，请注意，如果您在 nslookup 的查询画面当中，输入 set type=any 或其它参数， 那么就无法再进行反解的查询了！这是因为 any 或者是 mx 等等的标志都是记录在正解 zone 当中的缘故！

dig

```
语法：
[root @test root]# dig [@server] [-t type] [FQDN]
参数说明：
-t type ：查询某主机的某个卷标，例如 MX/NS 等等，以及所有标签 any 等
@server ：如果不想以 /etc/resolv.conf 来作为 DNS 主机，则可在此填入
          其它的 DNS IP！
范例：

[root@test root]$ dig @192.168.1.2 mdk.vbird.tw
; <<>> DiG 9.2.3 <<>> @192.168.1.2 mdk.vbird.tw
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40211
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;mdk.vbird.tw.                  IN      A

;; ANSWER SECTION:
mdk.vbird.tw.           600     IN      A       192.168.1.2

;; AUTHORITY SECTION:
vbird.tw.               600     IN      NS      mdk.vbird.tw.

;; Query time: 4 msec
;; SERVER: 192.168.1.2#53(192.168.1.2)
;; WHEN: Sun Oct 31 12:24:00 2004
;; MSG SIZE  rcvd: 60
# 我用我的 DNS 主机 192.168.1.2 去查询 mdk.vbird.tw 这个主机，
```

```
# 可以得到 A 与 NS 的结果！与 host -a mdk.vbird.tw 是否很类似啊！

[root@test root]$ dig @192.168.1.2 -t mx mdk.vbird.tw
; <<>> DiG 9.2.3 <<>> @192.168.1.2 -t mx mdk.vbird.tw
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15056
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;mdk.vbird.tw.                  IN      MX

;; AUTHORITY SECTION:
vbird.tw.               600     IN      SOA     mdk.vbird.tw.
root.mdk.vbird.tw.  2004102901 28800 14400 720000 86400

;; Query time: 4 msec
;; SERVER: 192.168.1.2#53(192.168.1.2)
;; WHEN: Sun Oct 31 15:51:29 2004
;; MSG SIZE  rcvd: 71
# 在这个查询方面，则主要在查询正解 zone 的 MX 标志。
```

事实上， host 与 dig 的查询输出结果挺类似的，而且 dig 的输出信息更多， 如果您的 DNS 发生不明原因的设定错误问题，可以利用 dig 的相关功能来查询喔！

whois

```
语法：
[root@test root]# whois domainname
范例：

[root@test root]# whois redhat.com
Registrant:
Red Hat, Inc. (REDHAT-DOM)
   P.O. Box 13588
   Research Triangle Park, NC 27709
   US

   Domain Name: REDHAT.COM

   Administrative Contact, Technical Contact:
      Network, Operations  (22271962I)          noc@REDHAT.COM
      Network Operations Center
```

```
     Red Hat, Inc.
     1801 Varsity Drive
     Raleigh, NC 27606
     US
     919-754-4177 fax: 919-754-3704

  Record expires on 25-May-2006.
  Record created on 26-May-1994.
  Database last updated on 31-Oct-2004 02:57:02 EST.

  Domain servers in listed order:

  NS3.REDHAT.COM              66.187.229.10
  NS2.REDHAT.COM              66.187.224.210
  NS1.REDHAT.COM              66.187.233.210
这个指令可以检视注册这个 domain name 的账号数据，
例如上面的例子当中，就是以红帽公司的领域名称来查询注册者的数据的啊！
```

whois 这个指令可以查询到当初注册这个 domain 的使用者的相关信息。不过， 由于近年来很多网络信息安全的问题，这个 whois 所提供的信息真的是太详细了， 为了保护使用者的隐私权，所以，目前这个 whois 所查询到的信息已经不见得是完全正确的了～ 而且，在显示出 whois 的信息之前，还会有一段宣告事项的告知呢～ ^_^y

无论如何，我们都可以透过 nslookup, host, dig 等等的指令来查询主机名称与 IP 的对应， 这些指令的用法可以请您以 man command 来查询更多的用法喔！

进阶设定：

其实， DNS 主机的运作原理与架设方式的变化，真的很高深莫测的！在这里， 我们额外的提出一些比较进阶的内容给大家参考参考，例如子网域的授权问题， 以及架设一个合法授权的 DNS 主机。

子网域授权问题

好了，那么万一我的网络很大，我只负责上层的 DNS 而已， 下层希望直接交给各单位的负责人来负责，要怎么设定呢？ 举个例子来说，以成大为例，成大计中仅管理各个系所的的主机 IP 而已，由于各个系所的主机数量可能很大，如果每个人都要请计中来设定， 那么管理员可能会疯掉，而且在实际设计上也不太人性化。所以啰，计中就将各个 subdomain (子网域) 的管理权交给各个系所的主机管理员去管理，如此一来， 各系所的设定上面会比较灵活，且上层 DNS 主机也不用太麻烦呐！

好了，那么如何开放子网域授权呢？我这里仅说明正解的部分，因为在 ISP 方面通常已经帮我们搞定反解了，所以先不理他！而目前我们去注册的信息上面， 通常仅取得的是正解的设定权，例如鸟哥的个人网域 vbird.idv.tw 就是一个例子！好了， 现在假设在我的 mdk.vbird.tw 上面，要将 win2k.vbird.tw 这个子网域切割出去给 win2k.vbird.tw 管理，那么该怎么办呢？

1. 主机端 mdk.vbird.tw 的设定：
   主机 mdk.vbird.tw 的设定其实很简单啦！只要将子网域开放出来给别人使用就对了！ 怎么设定呢？您可以直接修改 /var/named/named.vbird.tw ，使他变成如下所示：

```
[root@test root]# cd /var/named
[root@test named]# vi named.vbird.tw
# 再次提醒，这个文件名称是在 /etc/named.conf 里面设定
$TTL    600
@       IN      SOA     mdk.vbird.tw.       root.mdk.vbird.tw.   (
                        2004100601      ; Serial
                        28800           ; Refresh
                        14400           ; Retry
                        720000          ; Expire
                        86400   )       ; minimum
; 主机的设定参数部分
@                  IN      NS      mdk.vbird.tw.
@                  IN      MX 10   mdk.vbird.tw.
mdk                IN      A       192.168.1.2
mdk                IN      TXT     "The testing DNS server"
phorum             IN      CNAME   mdk
www                IN      CNAME   mdk.vbird.tw.
; 子网域分割出去给其它主机管理的例子！
win2k.vbird.tw.    IN      NS      win2k.vbird.tw.
win2k              IN      A       192.168.1.100
; 本机上面其它主机的设定信息方面
winxp              IN      A       192.168.1.200
```

   上面的特殊字体就是最重要的地方啦！我将 win2k.vbird.tw. 这个网域的 NS 权限(name server) 转给 win2k.vbird.tw 这部主机来管理，而底下列出来 win2k.vbird.tw 这部主机的正解信息！那么未来当有人要查询类似 www.win2k.vbird.tw 时，则先会到 mdk.vbird.tw 来查询， 而查到 win2k.vbird.tw 的网域，因此就会向下游的 DNS 亦即是 win2k.vbird.tw 这部机器查询了！

2. 下游主机 win2k.vbird.tw 的设定：
   这个设定就简单啦！直接参考一下我们上面写的数据，跟着设定，但是您的 domain name 变成 win2k.vbird.tw 就是了！简单的很呐！所以我就不再多说了～

架设一个合法的授权的 DNS 主机：

好啦！现在您应该知道什么是『经上游授权的合法 DNS 主机』了吧？！ 没错！就是上游的 DNS 主机将子网域的查核权开放给您来设定就对啦！嗯！虽然知道原理， 但是那么我要如何来架设一个合法的 DNS 主机呢？好让我自己管理自己的 domain ！举例来说，鸟哥的 vbird.idv.tw 就是 VBird 自己管理的哩～底下我们就来谈一谈，如何向 ISP 申请一个合法授权的 DNS 主机，或者是合法的主机名称啊！

1. 申请一个合法的 domain name

   既然是要建立一个合法的 domain name server ，自然就要向合法的 DNS 主机申请授权啰！ 目前您可以到底下的地方去申请喔！

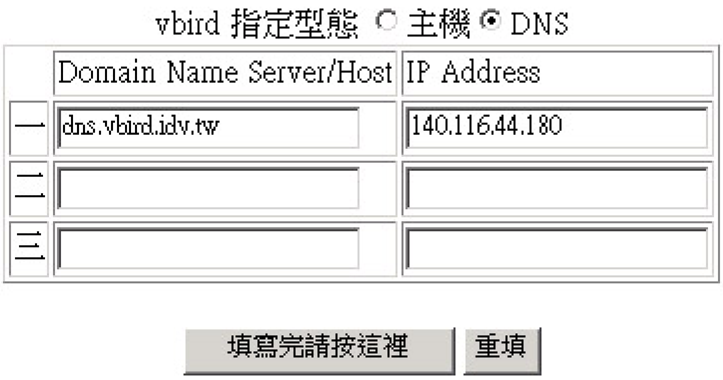
   - http://www.twnic.net/index3.php

   其实台湾地区的一些 domain 已经不再于 TWNIC 受理了，所以您连上上述的网站之后， 可以点选里头相关的连结到各大 ISP 去注册！例如鸟哥就注册了 vbird.idv.tw 这个网域！ 现在鸟哥就以 Hinet 的注册做为说明吧

   - 进入主画面：直接连结到底下的网页去： http://nweb.hinet.net
   - 选择需要的网域名称，并查询该网域是否已存在： 因为网域必需是独一无二的，所以您必需使用该网页当中提供的查询功能， 去查询一下您想要的网域是否已经被注册了呢？ 一定要没有被注册的网域才可以喔！
   - 逐步进行注册： 然后以该网站提供的功能一步一步的往下去进行，例如以鸟哥的『个人网址』之注册为例， 按下个人网址之后，会出现流程步骤为：



   - 选择网站代管或架设 DNS 模式：还记得前面提到的观念吧？对啦！我们可以直接请 ISP 帮我们设定好 host 对应 IP 就好(最多三部)，当然也可以自行设定一下我们所需要的 DNS 主机啦！如果未来您可能会架设 mail server ，所以还是自行设定 DNS 主机好了！选择上面图示的第五项『DNS 指定/异动』项目， 会出现下面图示。记得选择『DNS』及填写您的 hostname 与正确的 IP 即可喔！注意： 要填选这个项目，最好您的 IP 是固定制的，浮动制的 IP 不建议用这个选项！



- o 注册完毕！

2. 以 DNS 主机的详细设定 之设定内容来设定您的主机：
   如果您已经以 DNS 主机的方式申请了一个 domain name ，那么您就必须要设定您的 DNS 主机了！ 请注意，这个情况之下，您只要设定您的注册的网域的正解即可！ 反解部分则先不要理会，当然，如果您有办法的话，最好还是请上层的 ISP 帮您设定啰！
3. 测试：
   如此一来，您的 DNS 主机上面设定的任何信息，都可以透过 Internet 上面的任何一部主机来查询到喔！够棒吧！心动了吗？赶快去试看看吧！ ^_^

LAME Server 的问题：

如果您是架设 DNS 主机的新手，那么『一定』会在 /var/log/messages 这个登录档案里面发现到类似这样的讯息：

```
[root@test root]# more /var/log/messages
Oct  5 05:02:30 test named[432]: lame server resolving
'68.206.244.205.in-addr.arpa' (in '206.244.205.in-addr.arpa'?): 205.244.200.3#53
Oct  5 05:02:31 test named[432]: lame server resolving
'68.206.244.205.in-addr.arpa' (in '206.244.205.in-addr.arpa'?): 206.105.201.35#53
Oct  5 05:02:41 test named[432]: lame server resolving
'68.206.244.205.in-addr.arpa' (in '206.244.205.in-addr.arpa'?): 205.244.112.20#53
```

这是什么东西呐？！根据官方提供的文件数据来看 (在您的 Red Hat 9 的系统下，请察看这个档案『/usr/share/doc/bind-9.2.1/armBv9ARM.ch06.html』)，当我们的 DNS 主机在向外面的 DNS 系统查询某些正反解时，可能由于对方 DNS 主机的设定错误，导致无法解析到预期的正反解结果，这个时候就会发生所谓的 lame server 的错误！

那么这个错误会让我们的 DNS 主机发生什么严重的后果吗？既然仅是对方的设定错误， 所以自然就不会影响我们的 DNS 主机的正常作业了。 只是我们的 DNS 主机在查询时，会发生无法正确解析的警告讯息而已， 这个讯息虽然不会对我们的 Linux 主机发生什么困扰，不过，对于系统管理员来说， 要天天查询的

/var/log/messages 档案竟然有这么多的登录信息，这是很讨厌的一件事！

好了，我们知道 lame server 是对方主机的问题，对我们主机没有影响， 但是却又不想要让该讯息出现在我们的登录档 /var/log/messages 当中， 怎么达到这样的功能呢？呵呵！就直接利用 BIND 这个套件所提供的登录档参数啊！ 动作很简单，在您的 /etc/named.conf 档案当中的最底下，加入这个参数即可：

```
1. 修改 /etc/named.conf
[root@test root]# vi /etc/named.conf
// 加入底下这个参数：
logging {
        category lame-servers { null; };
};
// 注意一下，那个 logging 是主要的参数，至于 category 则是定义出什么信息，
// 因为我们不要 lame server ，所以选择 lame-servers 这个参数，并定义
// 参数值为 null (空的的意思)，这样就修改完成了！

2. 重新启动 bind
[root@test root]# /etc/rc.d/init.d/named restart
```

记得重新启动 named 之后，还是要察看一下 /var/log/messages 喔！ 以确定 named 的正确启动与否！然后，嘿嘿，以后就不会看到 lame server 咯！

解决 rndc key 的问题：

由于 BIND 提供了比较安全的 BIND 管理机制，因此，比较新的 BIND 9 以后的版本， 都需要提供所谓的 rndc key ，才能正常无误的启动 Bind 喔！

那么如何提供 BIND 这个 Key 呢？很简单，只要执行 rndc-confgen 就行了！

```
[root@test root]# rndc-confgen
# Start of rndc.conf
key "rndc-key" {
        algorithm hmac-md5;
        secret "l6zE+CnSFuteQHxYwIGQqq==";
};

options {
        default-key "rndc-key";
        default-server 127.0.0.1;
        default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow list as needed:
# key "rndc-key" {
```

```
#       algorithm hmac-md5;
#       secret "16zE+CnSFuteQHxYwIGQqq==";
# };
#
# controls {
#       inet 127.0.0.1 port 953
#               allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
```

接下来，只要将上表复制到 /etc/rndc.conf ，并且将

```
key "rndc-key" {
      algorithm hmac-md5;
      secret "16zE+CnSFuteQHxYwIGQqq==";
};
controls {
      inet 127.0.0.1 port 953
              allow { 127.0.0.1; } keys { "rndc-key"; };
};
```

复制到 /etc/named.conf 就可以了！什么时候需要提供这个 Key 呢？ 如果妳启动 DNS 的时候，老是在 /var/log/messages 里面发现这一行：

couldn't add command channel 127.0.0.1#953: not found

这就表示您的 DNS 系统必须要提供这个 key 啦！请按表操课吧！

架设动态 DNS 主机：

谈完了上面这么多的设定之后，接下来，我们谈一个比较有趣的咚咚～ 那就是 动态 DNS 主机 的设定～

什么是动态 DNS 主机呢？还记得我们在 合法的 DNS 主机 里面提到的，如果我们本身是以拨接制的 ADSL 连上 Internet 的时候，基本上， 我们的 IP 是 ISP 随意提供的，因此每次上网的 IP 都不固定，所以，我们没有办法以上面的 DNS 设定来给予这种连上 Internet 的方法一个适当的主机名称。 也因此，如果我们想要利用这种没有固定 IP 的联机方法架设网站时，就得要有特殊的管道了～ 其中之一的方法就是利用 Internet 上面已经提供的免费动态 IP 对应主机名称的服务！ 例如之前提到的 http://www.adsldns.org (但是在 2004/10 月份中旬，这个网站挂点好久好久...) 或者是 http://www.no-ip.org 等等。

提供这样的服务利用的是什么原理呢？基本上， DNS 主机还是得要提供 Internet 相关的 zone 的主机名称与 IP 的对应数据才行，所以，动态 DNS 主机 (Dynamic DNS, 底下我们称为 DDNS 主机) 就必须要提供一个机制，让客户端可以透过这个机制来修改他们在 DDNS 主机上面的名称与 IP 对应数据才行。

我们的 BIND 9 也有提供类似的机制喔！那就是利用 update-policy 这个选项，配合认证用的 key 来进行数据文件的更新。简单的说， 1) 我们的 DDNS 主机先提供 Client 一把 Key (就是认证用的数据， 你

可以将他想成是账号与密码的概念）， 2）Client 端利用这把 Key ，并配合 BIND 9 的 nsupdate 指令，就可以连上 DDNS 主机，并且修改主机上面的 Zone file 内的对应表了。感觉上很像很简单喔！ 没错啊！架设上真的很简单的～底下我们就来尝试设定一下喔：

---

1. DDNS Server 端的设定：

   如同上面说的，我们必须要提供 client 一把认证用的 key ，那么这把 key 怎么产生呢？ 又，如何设定 DDNS 主机呢？这里提供一个案例。

   假设我有一部机器，主要是用来作为 WWW 主机用的， 但是没有固定 IP，而我已经有 mdk.vbird.tw 这个 DNS 主机了。假设我的这部 WWW 主机想要的主机名称是 web.vbird.tw 这个主机名称，那该如何设定呢？

   产生认证用的 key

   我要产生一把 key 给 web 这个主机使用 (领域名称为 vbird.tw)，可以这样做：

```
[root@test root]# mkdir -p /var/named/keys; cd /var/named/keys
[root@test keys]# dnssec-keygen -a HMAC-MD5 -b 128 -n HOST web
# -a [密码演算方法]：这个参数后面可以接几种不同的密码演算方式，
#                    包括 RSAMD5, RSA, DSA, DH 与 HMAC-MD5 ，
#                    在这里，我直接使用比较常用的 HMAC-MD5 这个算法～
# -b [加密字节]  ：加密的密码长度也是可以控制的！我只用 128 位而已。
# -n [拥有类型]    ：这把 Key 是给 Client 用来作为 HOST 更新或者是整个
#                    ZONE 的更新？一般来说，我们仅允许客户端更新他自己
#                    的主机对应，所以只要给予 HOST 的权限就好了！
# name             ：最后给予这把 Key 一个名称，我这里将这把 Key
#                    名称订为 web
Kweb.+157+29323
[root@test keys]# ls -l
total 8
-rw-------   1 root root 47 Oct 31 20:09 Kweb.+157+29323.key
-rw-------   1 root root 81 Oct 31 20:09 Kweb.+157+29323.private
# 最后会产生两个档案，分别是 Kxxx.key 及 Kxxx.private，
# 其中， .key 是主要用来作为 DDNS 主机端设定的公钥，这把 Key 的内容
# 会被复制到 /etc/named.conf 里面进行设定喔！
[root@test keys]# more Kweb.+157+29323.key
web. IN KEY 512 3 157 gxHUFoGGRE91YyAVuMBh+Q==
# 特别注意，上面输出的特殊字体中，就是 Key 的内容！
# 这些数据是经过加密的，最后会被复制到 /etc/named.conf 里面喔！
```

   如此一来，这把 Key 就被设定好了！其中，那个 *.key 的档案内容关于密码的部分， 会被设定于 /etc/named.conf 里面喔！

设定 DDNS 主机上的 named.conf 档案

在设定上面也是挺简单的，只要将 Key 的数据输入进去，再将要开放写入的 zone 设定好 policy (规则) 就好了！

```
[root@test keys]# vi /etc/named.conf
....(略)....
key "web" {
          algorithm hmac-md5;
          secret "gxHUFoGGRE91YyAVuMBh+Q==";
};
zone "vbird.tw" {
        type master;
        file "named.vbird.tw";
        update-policy {
                grant web name web.vbird.tw. A;
        };
};
....(略)....
# 注意到上头的 gran web name web.vbird.tw. A; 那一行，
# gran 后面接的就是 key 的名称，也就是说，我这把 web 的 key
# 在这个 zone (vbird.tw) 里面可以修改主机名称 web.vbird.tw
# 的 A 的标志，亦即是修改主机的 IP 对应啦！语法也就是：
# gran [key_name] name [hostname] 标签
# 也就是说，我的一把 key 其实可以给予多种权限喔！看您如何规范了。
```

设定好之后，给他重新启动，然后观察一下 /var/log/messages 里面有没有错误即可！ 如此一来，DDNS 主机端就设定妥当啰！

---

2. Client 端的更新：

   接下来则是 DDNS Client 端的更新了。首先，您必须要由 Server 端取得刚刚建立的那两个档案，请将刚刚建立的 Kweb.+157+29323.key 及 Kweb.+157+29323.private 利用 SSH 的 sftp 传送过来， 我将他放置到 /usr/local/ddns 里面去，然后测试看看：

```
[root@test keys]# mkdir /usr/local/ddns; cd /usr/local/ddns
# 假设此时您已经将两个档案给他复制到此目录下了！
[root@test keys]# nsupdate -k Kweb.+157+29323.key
> server 192.168.1.2
> update delete web.vbird.tw
> update add web.vbird.tw 0 A 192.168.1.222
> send  最后在此按下 [ctrl]+D 即可
# 请注意到 update add web.vbird.tw 0 A 192.168.1.222 这行，
```

```
# 他的意义说的是，新增一笔数据， ttl 是 0 ，给予 A 的标签，
# 对应到 192.168.1.222 的意思～
# 至于 nsupdate -k 后面加的则是我们在 Server 端产生的那个 key 档案！
```

然后您就会发现到 /var/named 里面多出一个暂存盘，那就是 /var/named/named.vbird.tw.jnl
当然，/var/named/named.vbird.tw 就会随着 client 端的要求而更新数据喔！

由于手动更新好像挺麻烦的，我们就让 Client 自动更新吧！利用底下这个 script 即可！

```
[root@test root]# vi /usr/local/ddns/ddns_update.sh
#!/bin/bash
#
# Update your Dynamic IP by using BIND 9 's tools
#
################################################
# History
# 2004/10/27     VBird    First time release
#
###############################################
PATH=/sbin:/bin:/usr/sbin:/usr/bin
export PATH

# 0. keyin your parameters
basedir="/usr/local/ddns"                # working directory
keyfile="$basedir"/"Kweb.+157+29323.key" # your ddns' key (filename)
ttl=600                                  # the ttl time ( 10 min. )
outif="ppp0"                             # Your interface (to internet)
hostname="web.vbird.tw"                  # Your hostname
servername="192.168.1.2"                 # The primary DNS server

# Get your new IP
newip=`ifconfig "$outif" | grep 'inet addr' | \
        awk '{print $2}' | sed -e "s/addr\://"`
checkip=`echo $newip | grep "^[0-9]"`
if [ "$checkip" == "" ]; then
        echo "$0: The interface can't connect internet...."
        exit 1
fi

# create the temporal file
tmpfile=$basedir/tmp.txt
cd $basedir
echo "server $servername"                        >  $tmpfile
```

```
echo "update delete $hostname A "                  >> $tmpfile
echo "update add    $hostname $ttl A $newip"      >> $tmpfile
echo "send"                                        >> $tmpfile

# send your IP to server
nsupdate -k $keyfile -v $tmpfile
```

您只要将上述的程序里面，特殊字体的部分给他修改一下，就能够以 /etc/crontab 的方式在您的系统内自动执行了！

利用 BIND 9 所提供的这个服务，我们只要具有一组固定的 IP ，并向 ISP 申请一个合法授权的 domain name， 就可以提供不论是固定或者是非固定的 IP 使用者，一个合法的主机名称了！ 并且，使用者也可以自行透过 nsupdate 来修改自己的 IP 对应！以让自己的主机 IP 永远与主机名称保持正确的对应！这对只有拨接制上网的用户来说，真是方便啊！

重点回顾：

- 在 Internet 当中，任何一部合法的主机都具有独一无二的主机名称，这个主机名称包含了 hostname 与 domain name ，并称为 Fully Qualified Domain Name (FQDN)；
- 为了克服人类对于 IP 不易记忆的困扰，而有名称解析器的产生，首先是 /etc/hosts ，而后则是 DNS 系统的产生；
- DNS 服务器的类型主要分为 master, slave 以及只进行快取记录的 cache-only 的 DNS 主机；
- Master/Slave 架构下的 DNS 主机系统，不论是 Master/Slave 主机均需要能够正确的提供 hostname 与 IP 的对应才行。
- Slave 主机本身并没有自行设定 zone file ，其 zone file 是由 Master 主机传送而来，因此，master 主机必须要针对 slave 主机开放 allow-transfer 的设定项目才行。
- 目前 Unix Like 的机器当中，都是以 BIND 这个柏克莱大学发展的 DNS 套件；
- 在 DNS 系统中，正解为由 hostname 找 IP ，而反解则是由 IP 找 hostname ，至于 zone 则是一个或者是部分网域的设定值；
- bind 的设定档为 /etc/named.conf ，而 named.conf 可以规范出正反解 zone 的档案所在；
- 正解的纪录(record)主要有：SOA, A, MX, NS, CNAME, TXT 及 HINFO 等；
- 反解的纪录主要有： SOA, PTR 等；
- 在 client 端设定 DNS 查询顺序与相关功能的几个重要档案为： /etc/nsswitch.conf, /etc/hosts, /etc/resolv.conf 等；
- DNS 查询的指令主要有： host, nslookup, dig, whois 等等；
- 在载入了 named 这个 daemon 之后，请务必前往 /var/log/messages 察看此 daemon 的成功与否。

本章与 LPI 的关系：

在 LPI 网站 http://www.lpi.org 里面提到的，关于 NFS 的考试题库的地方，只有在 LPI level 1 的 102 ，里面的 topic 113 Networking Services ，第五点当中，简易的 DNS 设定。强调的是『应试者需了解何谓正、反解、Zone 与 cache-only 的 DNS 主机』至于会考的档案与指令可能有这些：

- /etc/hosts
- /etc/nsswitch.conf
- /etc/resolv.conf
- /etc/named.boot(V4)及 /etc/named.conf(V8)
- named (这个 daemon)

参考资源：

- BIND 官方网站：http://www.isc.org/products/BIND/
- Study Area 学习网站：http://www.study-area.org/linux/servers/linux_dns.htm
- 优客笔记：http://turtle.ee.ncku.edu.tw/~tung/dns/dnsintro.html
- lame server 的简易说明：http://linux.cvf.net/lame_server.html
- DDNS 架设：http://www.study-area.org/tips/ddns.htm

本章习题练习：

- 为何要有 DNS 系统：
- 那么请教 Unix Like 系统当中，主要使用那个套件做为 DNS 主机的架设，同时，他又是使用那个 daemon 来启动 DNS 系统？
- 最早的 Internet 其实是为了政府人员可以连上网络以进行资源的分享，另外，则是电子邮件的使用。而在早期使用的重要档案只有 /etc/hosts 这个，请教这个 hosts 档案的内容含有什么项目？
- 请说明 DNS 的三种类型与相关的内容：
- 正解档案(forward)反解档案(reverse)与内部循环使用的档案(loopback)主要的纪录功能为：
- 在主要的 DNS 设定档 /etc/named.conf 当中，有一个较为特殊的档案，他的类型为 hint ，请问这个档案的功能为何？
- 在 client 端搜寻 HOSTNAME 对应到 IP 的查询时，最重要的档案，以及该档案的主要用途为何？
- 一般来说，在 Client 端使用的查询 HOSTNAME 的指令大多使用什么？
- 请问 named 重要的信息登录在在那个档案中？

简易 WWW 服务器设定

最近更新日期：2003/10/02

我们最常讲的『架站』其实就是架设一个 Web 网站啦！那么什么是 Web 呢？说穿了，就是全球信息广播的意思(World Wide Web)，或者也可以称之为互连网吧！这个是我们目前的人类最常使用的 Internet 的协议之一啦！通常说的上网就是使用 WWW 来查询使用者所需要的信息啰！^_^！目前的 WWW 服务器主要分为两大阵营，分别是 Unix-Like 上面的 Apache 与 Windows 上面的 IIS ，就以价格效能比来说，当然是 Linux 上面的 Apache 最棒啦！至于 WWW 服务器的类型可以分为静态与动态，而这些动态的网站里面，很多都是以目前的当红炸子鸡 Linux + Apache + MySQL + PHP 架设而成的，简称为 LAMP 的咯！这种动态 WWW 主机很有趣喔！他可以沟通 Server 与 Client 端的数据呢！赶紧来进入这个 LAMP 的世界吧！

：1. 关于显示中文的额外说明：

：2. 关于无法执行 PHP 的说明：

：3. 关于 MySQL 的问题说明：

：4. 关于启动 httpd 时出现的 perl 问题：

课后练习：

原理：什么是 WWW 与网址

WWW 是 World Wide Web 的缩写，其中， Web 有广播网的意思存在，所以，简单的说，WWW就是全球信息网，可以结合文字、图形、影像及声音等多媒体，并透过超级链接 （ HyperText ） 的方式，将信息透过 Internet 传递到世界各处！那 WWW 的数据是如何传递的呢？

如果你常上网浏览的话 （ 不论是使用 Mosilla 、IE 或是 Netscape ），你应该会知道，台湾有个蛮有名气的入口网站：奇摩雅虎站 （ tw.yahoo.com ），所以你只要在网址列上面输入『 http://tw.yahoo.com 』就可以浏览到奇摩雅虎的网页信息啰！这个『 http://tw.yahoo.com 』就是所谓的 URL（Uniform Resource Locator），其中 tw.yahoo.com 就是所谓的主机名称，亦即是我们前面刚刚提过的 DNS 里头的 FQDN（ Fully Qualified Domain Name），即是主机名称加上领域名称所得的一个独一无二的 Internet 上面之名字啦！由于计算机在网络里面仅认识 IP 而已，所以，奇摩雅虎站的真实 IP 就是透过 DNS 解析 tw.yahoo.com 而找到这部计算机，然后经过 WWW 的协议功能将数据传到你的眼前来！那么有哪些格式的 URL 呢？呵呵！整个来说，网址列可以输入这些咚咚：

<协定>://<主机地址>[:port]/<目录资源>

- 协定：包括 http, ftp, news, gopher, telnet 这几种常见的方法！ 其中呢，http 是利用『主机的 http port ，通常为 80 』，至于 ftp 这个方法则是利用『主机的 ftp port ，通常为 21 这个埠口』，请注意喔，我们使用的 80 与 21 都是主机所提供的服务喔！而不是我们 client 端的 port ㄋㄟ，所以，使用 http 与 ftp 连上同一部计算机，所取得的信息并不见得会一样，因为服务本身就不同嘛！一个是 WWW 一个是 ftp 怎么会相同。此外，如果你没有指定协议的话，那么预设的协议就需要看客户端 （Client）使用的那个联机程序的预设协议了！举个例子来说，如果你是使用 Netscape 的话，呵呵！那么预设的协议就是 http 啰，因此，你在网址列输入 tw.yahoo.com 时， Netscape 立刻就会以 http 来连接出去啰！ ^_^;

- 主机地址：刚刚才提过的 FQDN 应该还没有忘记吧！由于计算机仅认识 IP ，所以，如果你输入 IP 的话，同样的可以联机喔！但是，如果是输入主机名称 （ domain name ） 的话，那么你就必需要让该主机名称可以经由转译器得到对应的 IP 喔！ 转译器是什么？就是 /etc/hosts 或者是 /etc/resolv.conf 里面的设定啊！当然，对外提供正常的 WWW 服务时，你的 host name 就必需要让大家可以转译到 IP ，就需要去申请一个合法的领域名称啰！ ^_^

- 目录资源：其实这个是 Uniform Resource Indicator, URI 的意思，如果你要去的网站网页在主网页的目录底下，那么你可以直接输入目录与网页的名称，就可以直接取得那个页面的数据啰！此外，如果你只输入网址而已呢？并没有输入网页名称呢？那么在

Server 端将会自动的判断 (看 Server 自己的设定而定) 该目录下是否有设定中引用的网页名称啰！这个在底下我们会再次的提到喔！

- :port：一般而言，各个协议都有其独特的使用的 port ，例如众所皆知的 http 使用的是 80 而 ftp 使用的是 21 这些个 port，所以，当你要连接到某个网站时，输入 http://that.host.name 就会主动的利用 80 那个 port 来尝试连接到对方主机！但是如果你不想要使用该 port 呢？举个例子来说，假如你的网站使用的是 8080 这个 port 来进行 WWW 的服务，果真如此的话，那么除非你有进行防火墙内的 port 对应，否则直接在网址列输入 http://your.host.name 结果将无法连接到你的 WWW 服务器，因为他会主动的连接到 80 那个 port 呐！所以，我们就要告诉浏览器，要向 Server 要求服务的是哪一个 port ，因此，你就要将他写成： http://your.host.name:8080！才可以连接到对方的 8080 那个 port 喔！

举个例子来说，我们通常去到中山大学的 FTP 网站都是以 Web 接口进入的，因为可以直接以浏览与搜寻的功能去捉取数据，因此，这个时候我们可以在网址列输入：
『 http://ftp.nsysu.edu.tw 』以进入 WWW 界面的 FTP 网站！请注意，是 WWW 界面喔！那么如果我想要直接以 FTP 的模式来进行数据的浏览与传输呢？呵呵！直接在网址列输入以 ftp 为方法的网址：『 ftp://ftp.nsysu.edu.tw 』呵呵！是否发现两者显示的咚咚不太相同？！没错啦！那就是因为我们所连接的主机的协议不一样的缘故，所以主机响应的数据当然就不同啦！请特别留意这种网址列的格式喔！

Client 如何向 Server 要求数据：

那么 WWW 是透过什么样的协议来传达数据的呢？呵呵！没有看到网址列的 http 吗？对啦！ WWW 就是使用所谓的 http 这个协议来传送数据的，HTTP 即是 HyperText Transfer Protocol 的简写，亦即是目前 WWW 的资料传递主流协议啦！而在网站上面供人浏览的网页，则大部分需要符合 HyperText Markup Language (HTML) 的语法啰！也就是说，当我们在网址列输入主机的网址之后：

5. Client 端先经过 DNS 解析得到 WWW 主机的 IP ，然后会发出一个数据封包，以 http 这个协议(或方法)联系到 WWW 主机，告知 WWW 主机我们要以 http 的方法来取得数据，同时，这个时候使用的是 TCP 协议，亦即需要经过三向交握的过程喔；
6. WWW 主机收到这个数据封包之后，会根据 Client 端的要求，提供相关的讯息来响应，大部分的情况下皆是使用 http 的协议传送具有 HTML 语法的网页数据到 Client 端的浏览器上；
7. 最后 Client 端的浏览器将 HTML 的语法经过解析后，以相关的画面来显示到屏幕上，提供用户来观赏喔！

这就是主要的流程啦，不过，Client 传到到 WWW 主机，与 WWW 主机响应的讯息里面，可包含有哪些可能的动作呢？

- GET：这是最常见的，就是 Client 端向 WWW 主机要求的资源，也可以看成 Client 端向主机取得的数据；
- HEAD：主机端响应给 Client 端的一些数据文件头而已；
- POST：Client 端传送到 WWW 主机端的数据；
- OPTIONS：主机端响应给 Client 端的一些允许的功能与方法；
- DELETE：删除某些资源的举动。

大致上就有这些功能，当然啦，最主要的就是 GET 这个功能啦！毕竟我们连上 WWW 主机就是为了要取得他的数据嘛！ ^_^。要记住的是，因为未来我们可能会去分析网站上的数据，所以，你必需要了解一下什么是 GET 或 HEAD 等等的意思！不然很多东西很难理解喔！ ^_^

有哪些类型的 WWW 网站？什么是 LAMP？

刚刚前头我们提到的都是关于 Client 端相对于 Server 端求取数据方面的问题，那么再来要谈的，是『咦！到底有哪些主要的 WWW 主机操作系统与软件之搭配呢？』呵呵！问的好！在回答这样的问题之前，我们先来讨论一下， WWW 主机的主要类型好了。基本的类型我们可以分为两种：

- 仅提供使用者浏览的网站：这种类型的网站大多是提供『静态』的网页，或许有提供一些动画图示，但基本上就仅止于此啦！因为他仅提供你来浏览，Server 不需要与 Client 端互动，所以你可以到该网站上去浏览，但是无法进行进行数据的上传喔！目前主要的免费虚拟主机大多是这种类型，所以，你只要依照 HTML 的语法写好你的网页，并且上传到该网站空间上，那么你的数据就可以让大家浏览了！

- 提供与使用者互动接口的数据库网站：这类型的网站可就多采多姿啦！因为他提供了与使用者互动的数据库软件，因此，使用者可以依据主机提供的服务，来进行留言、数据上传、存取的服务。由于 Server 与 Client 是互动的，因此一个接口良好的『数据库软件』就相当的重要！因为他可以在线实时来更新使用者所传递的数据讯息！这方面的网站例如最简单的留言版、讨论区、phpBB 架站论坛、phpnuke 架站论坛、金流与物流的商业型网站等等，都是属于这种互动类型的主机喔！

呵呵！这么看起来的话，似乎动态网页比较精彩喔！因为他可以跟使用者互动，也就更增加 WWW 主机的可变性与灵活运用之性质！所以啰，有办法的话，当然是选择动态网站的 WWW 主机架设比较好啰！那么动态网页的 WWW 主机需要些什么呢？就如同刚刚我们提到的，最重要的是那个可以随时更新数据的『数据库软件』所提供的信息，来与使用者互动，因此，一定要有数据库软件喔！再来，只有数据库，没有存取接口来沟通 Server 与 Client 端的数据传递当然还是无法直接在 Web 接口上面存取数据库的内容啦！因此，我们还需要一个『网页程序语言』来进行这个接口的编写哩！当然啦，最主要的还有就是需要 WWW 运作的软件啦！所以你需要：

- WWW 运作的主要软件：(目前有 Apache 与 IIS 两大系统)
- 数据库软件：(例如 MySQL, MS 的 SQL, 及其它相关的数据库)

- 编写网页的网页语言：(例如 shell scripts, perl scripts, Java, PHP CGI 等等)

那目前有哪些主流的个人动态 WWW 主机系统呢？大致上可以分为两种：一种是 Windows 系统的 IIS + MS 的 SQL + ASP WWW 服务器，这种 WWW 主机架设上蛮容易的，不过由于 Windows 的某些特性，所以很容易被 Cracker 所破坏；另外一种则是 Linux 系统上面的 Apache + MySQL + PHP 的 WWW 服务器 (简称 LAMP) ，这种服务器架设上有一定程度的困难度，尤其在升级与维护的方面，但是运作妥当的话，他的硬件要求、性能、安全性等方面，则相对的较佳喔！我们这里本来就是练习 Linux 的嘛！因此底下鸟哥将针对 Linux 系统上面的动态 WWW 主机进行介绍，当然啦，主角是 WWW 套件的 Apache 啦！至于需要了解的是： PHP 与 MySQL 分别是两个独立于 Apache 的套件，因此要让 Apache 这个 WWW 软件能够启用 PHP 与 MySQL 的功能，就必需要启动 Apache 里面的 PHP 与 MySQL 的模块啦！首先，未能免俗的，我们还是得分别介绍一下 LAMP 里面各个小东西的说明：

- Apache :在 1995 年之前就有蛮多的 Web 架设服务器软件的出现,不过,真正到了 1995 年之后，由国际超级计算机应用中心 (National Center for Supercomputing Applications, NCSA) 主导并克服了一些 Web 主机的臭虫之后，才让这个 http 协议的 WWW 套件得到了更广泛的应用!而因为这个释出的版本是来自于一些臭虫的克服，因此，这个 WWW 套件被戏称为『 A patchy server 』，意思就是说，一个经过更新后的 Server 的意思!后来，因为要将名字确定下来，干脆就直接取其谐音，用『 Apache 』，这也就是我们要介绍的 WWW 软件啦!

- PHP：官方的说法为：『PHP is a tool that lets you create dynamic web pages. PHP-enabled web pages are treated just like regular HTML pages and you can create and edit them the same way you normally create regular HTML pages.』所以说， PHP 可简单的视为一种程序语言，可以用来设计留言版、讨论区、或聊天室等等的动态网页的咚咚！由于他具有免费、跨平台、易学及效率高等等的优点，目前算是很盛行的一种设计网页的咚咚啦！ (基本上，PHP是使用来设计网页的程序语言，当然其功能不只如此！你可以轻易的在市面上找到相关的书籍喔)

- MySQL：将官方网站上的翻译文件中这么说：『MySQL是一个真正的多使用者、多执行绪 SQL数据库服务器。SQL（结构化查询语言）是世界上最流行的和标准化的数据库语言。MySQL是以一个客户机/服务器结构的实现，它由一个服务器背景执行程序mysqld和很多不同的客户程序和库组成。SQL是一种标准化的语言，它使得储存、更新和存取信息更容易。例如，你能用SQL语言为一个网站检索产品信息及储存顾客信息，同时MySQL也足够快和灵活以允许你储存记录文件和图像。MySQL 主要目标是快速、健壮和易用。』简单一点来说，这个东西就是一个数据库软件啦！例如：你在设计讨论区的时候，由于讨论的文章会日渐增多，因此就会有所谓的数据库处理的情况，MySQL 的目的就是在处理你这些由客户端传送来的数据。当然，其功能还不只此，我这里仅说一些我们可能用到的咚咚！

SSL 与 CA 的认证机制：

我们在前头有提过关于 HTTP 使用在传输上面的协议仍然是以 TCP/IP 为准，他传输的时候是使用明码来传送的，也就是说，在 Internet 上面流窜的 WWW 数据，基本上，都是以没有加密过的形式在传送数据！那么，当有些有心人士，利用 TCP Listen 的功能，即可将 Internet 上面的数据封包捉下来进行解析，并可能进一步取得该数据封包内的信息！『嘎！这有什么了不起，不过就是 WWW 信息而已，又不像 SSH 这种远程联机服务器的重要！』嘿嘿！这您可就有所不知了。要晓得的是，我们的网站并不涉及金流及物流的信息，所以当然没有什么『隐密性』可言，但是，如果今天换成是一个交易网站呢？例如网络书店的信用卡交易，例如一些金融公司提供的网络交易行为！这些讯息当中，很多都是含有相当重要的私人讯息ㄋㄟ～例如信用卡、身份证等代表个人的证号。万一被人撷取，呵呵！那可不是闹着玩的！所以啰，这个时候就需要有『数据加密』的动作了！目前用在 WWW 上面的主要加密功能，有 Secure Socket Layer (SSL) 及 Certificate Authorities (CA)两个主要的模式。

Secure Socket Layer (SSL)
不晓得您是否还记得我们在 远程联机服务器 里面提到的关于 SSH 这个服务器的联机过程!？也就是利用 Server 提供的 Public Key 并配合 Client 端随机产生的 Private Key 来组成一组加密(Public Key)与解密(Private Key)的方法！呵呵！这个方法同样的也被运用于 WWW 主机的设定啦！而支持这个 WWW 主机进行 Public 与 Private 加密的套件，就是很多时候都被拿出来使用的 OpenSSL 这个好家伙了！所以啰，要让你的 WWW 具有 SSL 加密的功能，就必需要安装 OpenSSL 这个套件才成呐!基本上,当 Client 端要向 Server 端求取数据的时候,则利用 Server 端本身提供的 Public Key 及 Client 端随机产生的 Private Key 组成一组可供利用的密码组合！则数据由 Server 传送到 Client 端之前，会先经由 Server 的 Public Key 将数据封包加密，而到了 Client 之后，才经由 Private Key 将数据解密！所以，当数据在 Internet 上面跑时，他是加密过的数据封包喔！即使被人劫取下来，他不晓得 Public 与 Private ，那么要解密可能也得费上几天几夜，甚至是好几年的功夫ㄋㄟ！因此，数据就会比较安全啦！当然啦，以我们这种主要以分享为主的网站，自然不需要使用这种技术！反正数据本来就是 Open 的！ ^_^

Cerificate Authorities (CA)
CA 这个方法同样也是使用 Public 与 Private Key 的方式，（ 呵呵！我们可以说，目前加密与解密的行为大部分都是使用这种类似的观念来进行的啦！ ）由于 SSL 使用的 Public 是 WWW Server 自行建立与产生的，所以具不具有公信力我们还不是很清楚！万一你连上去的 WWW 网站是个骗人的集团建立的，那么有没有加密对你而言，不都是粉危险的吗？咦！那么是否可以透过第三公证人来查验这个 Server 的 Public Key 呢？呵呵！ CA 这个方法就是要达成这个目的啦！基本上， CA 是一个公认的合法组织，他可以用来查验 WWW Server 提供的 Public Key 是否合法！以保障 Client 者的权益。因此， CA 是要钱的喔！

我们这里对于 SSL 及 CA 这两个咚咚仅提及他的概念，底下的文章并没有提到要怎么制作！有兴趣的朋友得自行到相关的网站去查询喔：

- SSL ： http://www.modssl.org/
- CA 组织之一：https://digitalid.verisign.com/server/apacheNotice.htm

---

套件安装：

好了，终于提到了我们要安装的 LAMP 的地方啦！如何安装 LAMP 呢？基本上仍然是有两种方式的，一个是 RPM 另一个就是 Tarball 啦！详细的观念请参考 鸟哥的 Linux 私房菜--基础学习篇 里面的RPM 与 Tarball 这篇文章吧！请仔细的阅读喔！因为两种方法安装上的难易度差异性很大，而且众所皆知的， RPM 的档案与 (1)Linux 发行厂商与版本 (2)及其它相依套件 之间有很大的相依性，所以你不能随便的拿网络上得到的 RPM 档案来安装！但是 Tarball 又很难安装完整！唉～真是两难！底下我们分别以 RPM 与 Tarball 的安装来说明！在 RPM 方面，我们分别以 Mandrake 9.0 与 Red Hat 9 的 Apache + MySQL + PHP 来说明，其中 Mandrake 提供的是 Apache 1.3.xx 版本，至于 Red Hat 9 则是提供 httpd-2.xx 版本 (注：Apache 这个套件在 2.x 以后的版本当中，套件名称已经改为 httpd 了，原来的 1.3.xx 版本则是以 apache 为套件名称喔！)至于 Tarball 的方式，则都以最新的 Apache + MySQL 及 PHP (2003/09/10)来安装啦！如果您还是希望以 Apache 1.3.xx 版本来安装您的 LAMP 时，那么请参考这篇： LAMP 的安装方法(http://linux.vbird.org/linux_server0360apache-1.php)。底下的 Tarball 安装方式仅适合 Apache 2.xx 版喔！不要搞错啰！ ^_^

RPM 安装 LAMP 方式：以 Mandrake 9.0 与 Red Hat 9.0 为例

既然要安装 LAMP 自然需要 Linux 系统，以及 Apache, MySQL 及 PHP 啰！我们以 Mandrake 9.0 提供的操作系统，及预设的相关档案来安装起所需要的套件！需要的套件至少有：

- 相关的函式库安装：
  - libmml-1.1.3-10mdk.i586.rpm
  - libmml-devel-1.1.3-10mdk.i586.rpm
  - libmml-static-devel-1.1.3-10mdk.i586.rpm
- PHP 安装：
  - php-common-4.2.3-1mdk.i586.rpm
  - php-4.2.3-1mdk.i586.rpm
  - php-devel-4.2.3-1mdk.i586.rpm
  - php-imap-4.2.3-1mdk.i586.rpm
  - tllibl-1.3.1-6mdk.i586.rpm
  - php-gd-4.2.3-1mdk.i586.rpm
- MySQL 安装：
  - libmysql10-3.23.52-1mdk.i586.rpm
  - MySQL-client-3.23.52-1mdk.i586.rpm
  - MySQL-3.23.52-1mdk.i586.rpm
  - php-mysql-4.2.3-1mdk.i586.rpm
- Apache 安装：
  - apache-common-1.3.26-6mdk.i586.rpm
  - apache-conf-1.3.26-3mdk.i586.rpm
  - apache-modules-1.3.26-6mdk.i586.rpm
  - apache-1.3.26-6mdk.i586.rpm
  - apache-manual-1.3.26-6mdk.i586.rpm

- libgdbm2-devel-1.8.0-18mdk.i586.rpm
- libdbtcl3.3-3.3.11-11mdk.i586.rpm
- libdb3.3-devel-3.3.11-11mdk.i586.rpm
- db1-devel-1.85-8mdk.i586.rpm
- apache-devel-1.3.26-6mdk.i586.rpm
- mod_php-4.2.3-1mdk.i586.rpm
- mod_perl-common-1.3.26_1.27-7mdk.i586.rpm
- apache-mod_perl-1.3.26_1.27-7mdk.i586.rpm

档案可不少哪！所以需要好好的选择来安装才行！『请依照上面的顺序一个一个的安装下去吧！』当然，这么安装会疯掉的呐！所以我们建议使用 urpmi 来安装您的 Mandrake 的 LAMP 主机喔！请回到『网络升级套件』那一章节去查阅如何使用喔！

好了，那么如果是 Red Hat 9 呢？他需要的套件至少也需要底下这些咚咚呐：

- Apache 的相关套件
  - httpd-2.0.40-21.5
  - httpd-devel-2.0.40-21.5
  - httpd-manual-2.0.40-21.5
- MySQL 的相关套件
  - mysql-3.23.56-1.9
  - mysql-devel-3.23.56-1.9
  - mysql-server-3.23.56-1.9
- PHP 的相关套件
  - php-4.2.2-17.2
  - php-devel-4.2.2-17.2
  - php-mysql-4.2.2-17.2

当然还有很套件没有列出来咯，您可以使用 apt 去安装这些套件，以克服属性相依的问题呢！重点在 httpd, mysql-serve, php 以及 php-mysql 这几个套件说！挺重要的喔！赶快去安装吧！

鸟哥的特别告知：特别注意到，这里是以 Mandrake 9.0 与 Red Hat 9 做为介绍的，如果您不是以这个版本来安装你的 Linux 系统的话，而且您所在的环境并没有连上 Internet 时，那么在档案的名称上面可能会有一点点的不同！不过不要紧啦！因为你可以使用：

8. 直接 mount 你的 CDROM ，不会 mount 我可是会 K 人的喔！ ^_^：例如『mount -t iso9660 /dev/cdrom /mnt/cdrom 』；
9. 然后以 find 搭配 grep 来找出相关的 php, apache, 或 mysql 的字眼，注意到，mysql 有时候会有大写，有的套件仅要小写即可！都不太一样啦：例如『 find /mnt/cdrom -type f | grep -i php 』将数据都给他找出来啦！

10. 最后就一个一个的装上去，不过，这里会出现很多的问题，那就是各个套件的属性相依的问题啦！这个时候怎么办呢？没办法啦，就只有将需要的相依属性的档案一个一个的装上去啰！还要再找出来其它的档案来装的意思啦！ @_@

这里不谈完全的安装，我们仅安装可以让 Apache 跑 MySQL 及 PHP 等基本的套件，不玩 LDAP 及其它的咚咚！等到您对于 Apache 有一定的概念之后，自然就可以针对你希望的模块来加以设计了！ ^_^其中，最重要的是 mod_php (Mandrake 9.0) 或者是 libphp4.so (Red Hat 9)喔！如果你没有安装下去的话，那么很可能会造成 apache 无法执行 PHP 的困扰喔！OK！这样就安装完毕了！如果你不想要使用 Tarball 安装的话，可以跳到后续的 主机设定 去瞧一瞧设定的步骤啰！

Tarball 安装 LAMP 方式 (Apache 2.xx 2003/09/10 前)

Tarball 的安装方式方面，我们选择最新的套件来安装，如果您想要安装旧版的 1.3.xx 的 Apache 时，请参考 LAMP 的安装方法
(http://linux.vbird.org/linux_server0360apache-1.php) 一文。

下载各个套件：
要架设这样的一个主机需要哪些套件呢？不就是： Apache、MySQL及 PHP 啰！要从何处下载呢？你可以到中山大学的 FTP 站去搜寻，因为他提供的接口鸟哥真是蛮喜欢的，而且也真的很实用喔！寻找档案快又正确！

- 中山大学 FTP 站 (http://ftp.nsysu.edu.tw)

当然，你也可以到各个套件的发展处去下载：

- Apache：目前 Apache 已经出到了 2.0.X 版，但是最广泛使用的还是属于 1.3.X 版本。你可以上 Apache 主网页去看看相关的信息：
  - Apache 主页 (英文)：http://httpd.apache.org/
  - Apache 套件 (由主页下载)：http://httpd.apache.org/dist/httpd/
  - 台湾的映射站台：http://ftp.nsysu.edu.tw/Unix/Web/apache/httpd/

- PHP：目前最新的是 4.3.3 版 (2003/09/10) ，你可以上 php 的主网页去看看一些相关的咚咚喔！
  - PHP 主页 (英文)：http://www.php.net
  - PHP 主页下载：http://www.php.net/downloads.php
  - 台湾映射站台：http://ftp.nsysu.edu.tw/Unix/Web/php/

- MySQL：目前最新的版本是 3.23.57(2003/09/10)，同样的，你也可以上 MySQL 官方网页去看看喔!
  - MySQL 主页 (英文)：http://www.mysql.com/
  - MySQL 主页下载：http://www.mysql.com/downloads/mysql-3.23.html
  - 台湾映设站台：http://ftp.nsysu.edu.tw/Unix/Database/MySql/downloads/

也就是说，我们需要的档案有三个，分别是：

- httpd-2.0.47.tar.gz
- mysql-3.23.57-pc-linux-i686.tar.gz
- php-4.3.3.tar.gz

依序安装：
还记得我们在 Tarball 与 RPM 一文里面提到的 Tarball 的安装方法吗？没错！基本上，就是几个步骤而已：

- ./config (或 ./configuration) 建立 Makefile
- make 开始编译
- make install 开始安装到设定的目录去

大致上只有这样而已ㄋㄟ！简单吧！但是 Apache 的安装方法却不太简单！ Why ？？这是因为他还需要支持 PHP 这个玩意儿！所以就显的特别的麻烦～不过，现在有比较简单的方式了！呵呵！就是使用 Dynamic library 的方式 (动态函式库) 来安装 PHP ，哈哈！那么就不需要将他 compile 到 apache 里面去，而可以将他视作一个独立的模块！如此一来， PHP 的升级与安装就显的很简单啰！不过，由于动态函式库的安装虽然有好处，然而缺点就是....你的模块路径不能够随便乱摆！好在我们很少将编译好的模块随便移动的～呵呵！所以请注意：底下我们将 PHP 以『动态函式库』的形式来安装。

21. 先安装最简单的 MySQL ：
    为什么说 MySQL 最简单呢？这是因为在 官方网站 上面提到了一个问题，也就是使用 source code (Tarball) 的方式来编译时，如果您的 compiler (GCC) 版本高于 2.96 时，那么您所编译出来的 MySQL 程序，『有可能』会有数据库突然死掉的情况发生！因此， MySQL 的官方网站上面『建议』在目前的版本当中，最好直接以他们编译好的 MySQL 的 binary 版本来进行安装的动作！因为他已经帮你编译好了啊！所以，我们就不需要 make 啰!因此就变的很简单啦!此外，如果你的 Linux 版本中，你的 GCC 大于 2.96 时，且你使用的就是该 Linux 版本提供的 MySQL 时，官方网站上面，亦建议你直接将该 MySQL 移除，然后以他们的版本来安装，会比较没有问题啦！看来我们可能也需要升级一下 MySQL 啦！(无论如何， MySQL 仅是提出『有网友回报出有这个问题』，所以，如果您的 MySQL 向来就没有问题，那么就不用理会这个困扰了！)

另外，我怎么知道我目前的 GCC 版本呢？可以这样做：

```
1. 查询可以使用：
[root@test root]# rpm -qa | grep gcc
libgcc1-3.2-1mdk
gcc-cpp-3.2-1mdk
gcc-3.2-1mdk
gcc-c++-3.2-1mdk

2. 移除 MySQL 可以使用：
[root@test root]# rpm -e MySQL
```

22.

如果还不会使用 RPM ，那么就不要玩架站吧！说过好多次啰！ ^_^！上面的结果就显示我的 gcc 是 3.2 版，哇！太新了！比 2.96 版要更新的多！那么照 MySQL 官方网站的建议，还是置换成旧版本会比较好ㄋㄟ！如果要移除的话，那么就使用 -e 的参数来移除 MySQL 吧！(注：还是那句老话，请特别留意你的每个动作代表的意义，尤其是如果您的 MySQL 已经运作了一段时间了，请将 /var/lib/mysql 这个目录内的所有数据备份下来！) 好了！假设我已经将 mysql-3.23.57-pc-linux-i686.tar.gz 这个档案捉下来了，那么要如何安装呢？假设该档案在 /root 底下时：

```
0. 查询是否已经有 mysql 的账号：
[root@test root]# grep mysql /etc/passwd
# 如果没有 mysql 出现的话，那么请建立一个名为 mysql 的账号！
# 这个是要给 MySQL 的 Process 使用的！为了安全性，请务必建立！
# 如果之前已经建立过了，那么底下这一步建置的工作就可以跳过，
# 直接到 1. 解压缩与建立连结 去安装啰！

[root@test root]# groupadd -g 315 mysql
# 因为我刚好没有 315 这个 GID ，而 mysql 是系统使用的账号，我希望他在 500 以内，
# 因此就选择 315 做为 mysql 的 gid 啰！你当然可以变更这个数字，
# 使用小于 500 的 GID 做为系统的账号之用只是惯用的习惯而已啦！ ^_^

[root@test root]# useradd -u 315 -g mysql -d /usr/local/mysql/data -M mysql
# 我使用 315 做为 mysql 这个账号(与群组同名！)的 UID 啦！
# 并且建立他的家目录在 /usr/local/mysql/data 里面！

1. 解压缩与建立连结：
[root@test root]# cd /usr/local     <==因为已经是 binary 的套件，不用 make ！
[root@test local]# tar -zxvf /root/mysql-3.23.57-pc-linux-i686.tar.gz
...(讯息略过)....
```

```
# 最后会产生一个目录： mysql-3.23.57-pc-linux-i686

[root@test local]# ln -s mysql-3.23.57-pc-linux-i686 mysql
# 通常习惯将 MySQL 安装在 /usr/local/mysql 当中！但为了未来升级版本的确认，
# 官方网站上面建议使用连结的方式来进行 MySQL 的使用！

3. 档案权限修正：
[root@test local]# mkdir -p /var/lib/mysql
[root@test local]# chown -R mysql:mysql /var/lib/mysql
[root@test local]# chown -R root:mysql /usr/local/mysql-3.23*
[root@test local]# chown -R mysql:mysql /usr/local/mysql/data
# 修改成较为安全，且数据库所属人为 mysql 喔！特别留意啦！

4. 建立数据库：
[root@test local]# cd mysql
[root@test mysql]# ./scripts/mysql_install_db
[root@test mysql]# chown -R mysql:mysql /var/lib/mysql
[root@test mysql]# chown -R mysql:mysql /usr/local/mysql/data
# 这个步骤会在 /usr/local/mysql/data 里面建立好 MySQL 的数据库！
# 由于 /usr/local/mysql/data 是 MySQL 的数据库目录，所以很重要喔！请多加备份！
# 不过，在新版的 3.23.57 这个版本当中，数据库竟然移到 /var/lib/mysql 去了！
# 还真是有点奇怪呐！另外，根据诸多网友的回报，发现在建立数据库之后，
# 还需要重新设定一下数据库的所属群组与拥有者喔！

5. 启动测试：
[root@test mysql]# /usr/local/mysql/bin/safe_mysqld --user=mysql &
Starting mysqld daemon with databases from /usr/local/mysql/data
# 注意：这个时候 mysql 会建立一个 socket file 在 /var/lib/mysql/mysql.sock 喔！
# 未来我们在使用 MySQL 的各种指令功能时，都需要使用到这个 socket file，
# 但是 MySQL 偏偏预设的 socket file 是在 /tmp 底下，怎么办？！真讨厌，
# 我们可以透过这个简单的动作来欺骗我们的 MySQL 喔！
[root@test mysql]# ln -s /var/lib/mysql/mysql.sock  /tmp/
# 如果还是找不到 mysql.sock 时，请使用 find / -name mysql.sock
# 来找出这个档案的绝对路径吧！

[root@test mysql]# netstat -tl | grep mysql
tcp        0      0 *:mysql                 *:*                     LISTEN
[root@test mysql]# ps -aux | grep mysql
mysql     6394  0.0  1.5 10528  992 pts/3    S    16:16   0:00 /usr/local/mysql/
mysql     6395  0.0  1.5 10528  992 pts/3    S    16:16   0:00 /usr/local/mysql/
mysql     6396  0.0  1.5 10528  992 pts/3    S    16:16   0:00 /usr/local/mysql/
root      6422  0.0  1.1  2408  732 pts/3    S    16:20   0:00 grep mysql
# 呵呵！这样就应该是搞定了！ MySQL 已经在监听要求啰！而且所有人为 mysql ！
```

```
6. 开机后立即启动！
[root@test mysql]# vi /etc/rc.d/rc.local
# 将底下这一行加入这个档案的最后面一行喔！
cd /usr/local/mysql; /usr/local/mysql/bin/safe_mysqld --user=mysql &
# 这样一来，每次开机就可以自动的启动 MySQL 啰！
# 注：由于很多网友回复之问题中发现，如果没有加上 cd /usr/local/mysql 时，
# 会导致无法自动于开机的时候启动，因此，请大家记得加上这个动作呢！

7. 进阶设定内容：
#  由于我们 MySQL 放置的地点在 /usr/local/mysql 内，这个目录并不在 PATH 当中！
#  且 man page 亦不在 MANPATH 里面，所以，我们要手动的帮他加入啰！

[root@test mysql]# vi /etc/profile
# 大约在 33 行的地方，而且每个 distribution 设定的地方都不太相同！
# 请找到 export PATH ... 那一行，以 Mandrake 9.0 来说，大概在 33 行左右，
# 新加入一行：
PATH="$PATH":/usr/local/mysql/bin
export PATH ....(略)....

[root@test mysql]# vi /etc/man.config( 有的 distribution 为 /etc/man.conf )
# 可以在这个档案的任何地方加入底下这一行：
MANPATH /usr/local/mysql/man
# 就可以具有 man page 的能力了！

8. 建立 MySQL 的 root 账号密码！
[root@test mysql]# /usr/local/mysql/bin/mysqladmin -u root password 'your.password'
# 请建立密码！为了安全起见！否则你的 MySQL 数据库，将预设所有人都可以登入喔！
# 注意，如果执行上面的指令时，竟然出现如下的错误：
ERROR 2002: Can't connect to local MySQL server through socket '/tmp/mysql.sock' (111)
# 这表示 mysql 找不到 mysql.sock 这个档案！我们上面不是提到 mysql.sock 的
# 绝对路径吗？假设是 /var/lib/mysql/mysql.sock 好了，那么我们可以：
[root@test mysql]# /usr/local/mysql/bin/mysqladmin -u root \
>  -S /var/lib/mysql/mysql.sock password 'your.passwd'
# 当然也可以进行档案的连结阿！ ln -s /var/lib/mysql/mysql.sock /tmp

[root@test mysql]# /usr/local/mysql/bin/mysql -u root -p  \
> [-S /var/lib/mysql/mysql.sock] # 后面 [] 的内容不一定需要！且 [] 不要打！
Enter password: <==这里输入你刚刚建立的那个密码喔！
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.23.57

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> exit
```

```
Bye

# 这样就是可以确认已经可以连接到你的 MySQL 数据库了！请特别留意，有的朋友没有移除
# RPM 的 MySQL 时，那么您可能会有两个 mysql 的执行程序，一个在 /usr/bin/mysql ，
# 一个在 /usr/local/mysql/bin 里面，不要使用错档案，否则可能会显示如下的错误讯息：
ERROR 2002: Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock' (2)
# 解决的方法我们上面提过啰！ ^_^
```

23.

有这几个步骤就 OK 啦！您的 MySQL 已经建立了，而且管理者的账号 root 也已经建立的密码啦！请特别留意的是，这个 root 是 MySQL 的账号，与 Linux 在 /etc/passwd 里面的 root 是完全没有关系的喔！因为 MySQL 只是 Linux 里面的一个软件，任何 Linux 里面的使用者，只要知道 MySQL 的 root 密码，就可以使用 MySQL 的 root 功能！ 此外，特别需要留意的还有，由于我们使用的是 Tarball 的方式来安装的 MySQL ，所以我们的 data directory 会是摆在 /usr/local/mysql/data 当中！因此，有必要针对这个目录进行『备份』的工作！(注：还是得视您的 MySQL 版本来设定的！以 3.23.54a 版本来说，他确实是在 /usr/local/mysql/data ，不过 3.23.57 则改换到了 /var/lib/mysql 去了！真是伤脑筋呐！)千万不要大意备份的举动喔！

24. 再安装需要编译的新版 Apache 2.xx ：

因为目前有所谓的这个动态函式库，因此，我们在安装 Apache 的时候，请特别要『向 Apache 宣告 PHP 模块使用动态函式库的模式』来进行 PHP 的执行！这个时候，你可以这样的来安装你的 Apache 喔！

```
0. 解压缩：
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/httpd-2.0.47.tar.gz
# .....(讯息略)....
# 最后会产生一个 /usr/local/src/httpd-2.0.47 的目录

1. 搜寻设定内容：
[root@test src]# cd httpd-2.0.47
[root@test httpd-2.0.47]# ./configure --prefix=/usr/local/apache2  \
> --enable-so --enable-rewrite
# 上面请特别注意到：
 --prefix=/安装的路径：这个项目在设定未来你的 Apache 安装在那个目录当中？！
--enable-so        ：这个项目则是在宣告使用动态函式库啦！特别重要！
--enable-rewrite   ：这个项目只是预防用的！可以先设定，不过不一定会用到！
# 其它的额外项目请使用 ./configure --help 来察看吧！
# 按下 Enter 之后，会开始侦测你的主机内容！如果发生找不到 gcc 或 cc ，
# 那么肯定就是没有安装 make 或一些编译软件！请自行再加以安装吧！

2. 开始编译与安装：
```

```
[root@test httpd-2.0.47]# make; make install
# 如果没有错误的话，那么在 /usr/local/apache2 这个目录当中就已经将你的 Apache 安装了

[root@test httpd-2.0.47]# cd /usr/local/apache2
[root@test apache2]# ls -l
bin/         ：预设的 Apache 所有执行档案的放置目录
build/       ：一些编译过程中安装好的咚咚
cgi-bin/     ：预设的可以执行 CGI 的目录！！粉重要！
conf/        ：预设的 Apache 的参数文件放置的目录！！粉重要！！
error/       ：当使用者连上 server 有问题时，显示的错误网页在这里提供！
htdocs/      ：这个就是预设的主机的主页！！粉重要！
icons/       ：预设的一些小图示 ( icon ) 放置的目录
include/     ：其它一些 Apache 相关的函式库放置的目录
lib/         ：其它函式库放置的目录
logs/        ：登录讯息档案放置的目录喔！
man/         ：这个就是 man page 放置的目录
manual/      ：使用说明喔！
modules/     ：其它 Apache 使用的模块放置的目录！

3. 做个简易的修改：
#  奇怪的很，在 httpd.conf 这个 Apache 的设定档当中，竟然启用的 User
#  与 Group 有点怪怪的，所以这个时候我们必须要修正一下这个项目啦！
[root@test apache2]# vi /usr/local/apache2/conf/httpd.conf
# 找到底下这两行：
User nobody
Group #-1
# 粉奇怪吧！竟然是 #-1 那！而且 nobody 也不见得每部机器上面都有这个
# 系统账号，请查出您的 /etc/passwd 里面，是否有 nobody 这个账号，如果
# 没有 nobody 这个账号，可以使用 useradd -r nobody 来新增系统账号。同时，
# 查看一下您的 /etc/group 里面是否有 nobody 或者是 nogroup 的存在？
# 通常 Mandrake 会存在 nogroup 这个群组，至于 Red Hat 则会有 nobody 这个群组，
# 所以将上面两的项目改成底下的模样吧！
User nobody
Group nobody
# 然后储存后离开！

4. 确定启动状态：
[root@test apache2]# /usr/local/apache2/bin/apachectl start
[root@test apache2]# netstat -utl
tcp        0      0 *:http                  *:*                     LISTEN
# 呵呵！看到上面这行就表示您的 Apache 已经启动啰！当然啦！
# 有的人会看到的是：
tcp        0      0 *:www                   *:*                     LISTEN
# daemon 的名字会依照 /etc/services 而变呢！
```

```
# 而那个 apachectl 档案，就是启动的 scripts 啦！若要开机时启动 apache ，
# 那么将 /usr/local/apache2/bin/apachectl start 放在 /etc/rc.d/rc.local 内吧！

5. 进阶设定：
[root@test apache2]# vi /etc/profile
# 将刚刚我们上面 MySQL 时新增的一行，重新再改为如下所示：
PATH="$PATH":/usr/local/mysql/bin:/usr/local/apache2/bin

[root@test apache2]# vi /etc/man.config
# 再新加一行！
MANPATH /usr/local/apache2/man

6. 使用文字接口浏览器测试：
[root@test apache2]# lynx http://localhost
                                              Test Page for Apache Installation
   If  you  can see this, it means that the installation of the Apache web
   server  software on this system was successful. You may now add content
   to this directory and replace this page.
                     ____________________________________

              Seeing this instead of the website you expected?

   This  page  is  here  because  the  site  administrator has changed the
   configuration of this web server. Please contact the person responsible
   for  maintaining  this  server  with  questions.  The  Apache  Software
   Foundation, which wrote the web server software this site administrator
   is  using, has nothing to do with maintaining this site and cannot help
   resolve configuration issues.
                     ____________________________________

   The Apache documentation has been included with this distribution.

   You  are  free  to use the image below on an Apache-powered web server.
   Thanks for using Apache!
# 当然啦！要使用 lynx 必须先安装他！所以，请先将您的光盘 mount 然后安装 lynx 吧！
# 如果出现上面的测试网页，呵呵！恭喜您，您的 Apache2 已经可以正常的启动啰！
```

25.

为什么一定要在 /usr/local/src 底下进行 Tarball 呢？这仅是约定俗成的啦！因为如此一来，大家都安装在这个地方，以后主机的维护与移交都很简易！并且，对于您未来在主机上面的『升级』与『版本判别』都有很好的帮助呢！基本上，如果上面的过程都没有错误发生，那么恭喜您，已经可以顺利的来启动你的 Apache 了！并且应该有第一个主机网页产生啦！呵呵！真是快乐喔！但是如果有问题呢？通常最大的问题排除打字的错误之外，应该就是来自于我们在 Tarball 与 RPM 一文当中提到的，忘记安装那个

make, gcc 等套件！这个时候，请拿出您的光盘，一个一个的将套件安装上去吧！粉麻烦的啦！好了！接着下来看看怎么安装 PHP 咯！而这个方式安装下来的 Apache 首页就在 /usr/local/apache2/htdocs 这里啦！[注意：在这个安装的方法之后，很奇怪的是，并没有将中文的首页设定正确！你可以到 /usr/local/apache2/htdocs 里头，下达 cp index.html.zh index.html.tw.Big5 即可！]

26. 安装 PHP 在您的系统中：
    没有 PHP 时，您的网页将变的很单调ㄟ～而且粉多的 PHP 论坛都没能架设，很可惜！所以底下我们来谈谈怎么让你的 LAMP 正确的支持 PHP 啰！

```
0. 解压缩：
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/php-4.3.3.tar.gz
# .....(讯息略)....
# 最后会产生一个 /usr/local/src/php-4.3.3 的目录

1. 搜寻设定内容：
[root@test src]# cd php-4.3.3
[root@test php-4.3.3]# ./configure --prefix=/usr/local/php4   \
>--with-apxs2=/usr/local/apache2/bin/apxs   \
>--with-mysql=/usr/local/mysql   \
>--with-config-file-path=/usr/local/php4
# 上面请特别注意到：
--prefix=/安装的路径：这个项目在设定未来你的 Apache 安装在那个目录当中？！
--with-apxs2        ：这个则是 Apache2 专用的选项喔！请针对您的主机情况设定！
--with-mysql        ：这个则是针对 MySQL 啦！当然啦，就写我刚刚搞定的咚咚！
--with-config-file-path：这个又是什么？呵呵！是 php 的设定档 php.ini 放置的目录啦！
# 其它的额外项目请使用 ./configure --help 来察看吧！

2. 开始编译与安装：
[root test php-4.3.3]# make; make install
# 如果没有错误的话，那么在 /usr/local/php4 这个目录当中就已经将你的 php 安装好了！

3. 转存 PHP 基本组态档案：
[root@test php-4.3.3]# cp php.ini-dist /usr/local/php4/php.ini
# 这个路径与你刚刚在 ./configure 当中那个 --with-config-file-path 设定有关！

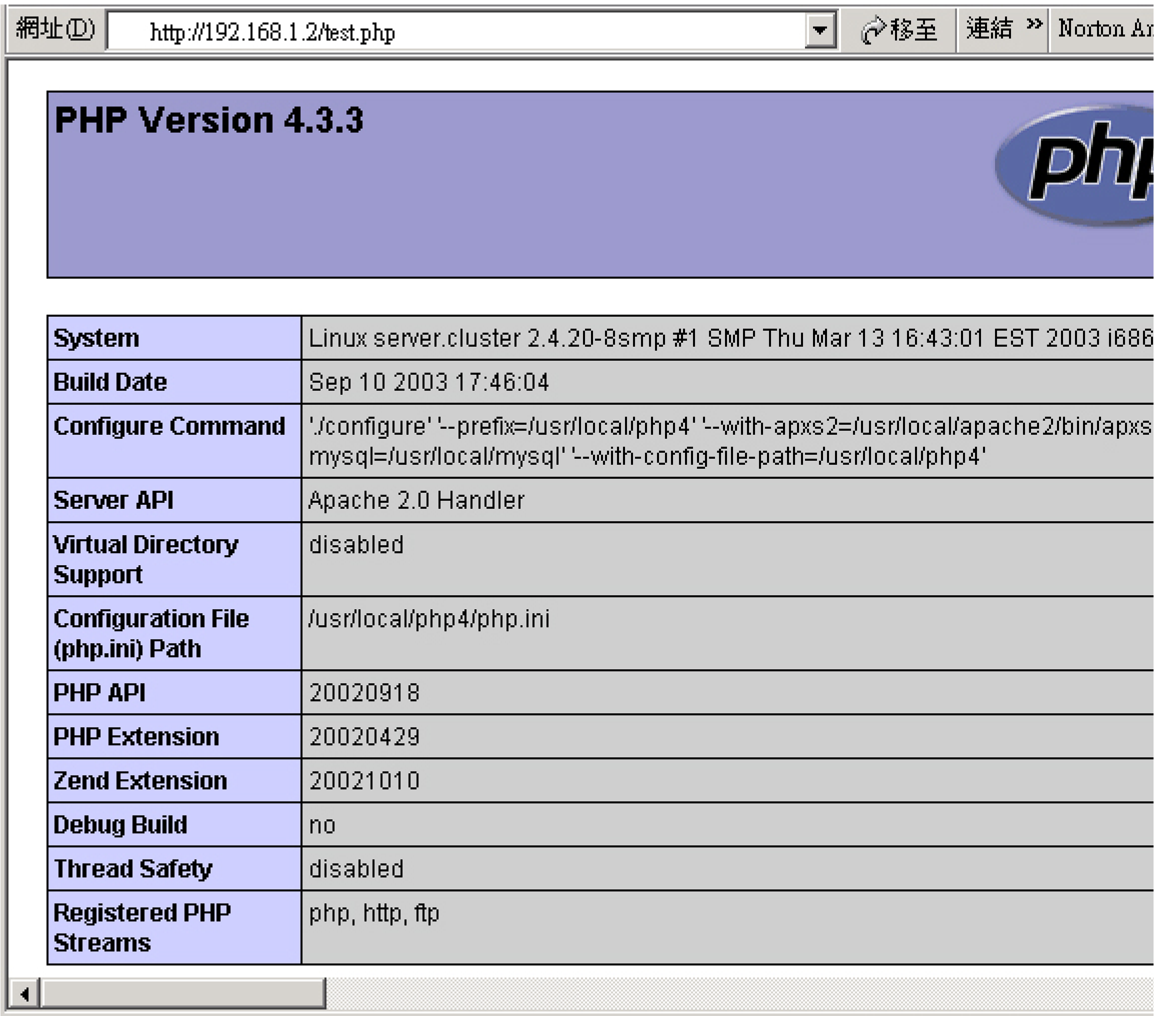
4. 启动 Apache 当中的 PHP 选项：
[root@test php-4.3.3]# vi /usr/local/apache2/conf/httpd.conf
# 找到底下两行：
   LoadModule php4_module modules/libphp4.so  <==大约在 231 行处
   AddType application/x-httpd-php .php      <==这一行可以在 847 行处自行增加！
```

```
5. 重新启动 Apache ：
[root@test php-4.3.3]# /usr/local/apache2/bin/apachectl stop
[root@test php-4.3.3]# /usr/local/apache2/bin/apachectl start

6. 测试 PHP 是否是正常工作的：
[root@test php-4.3.3]# cd /usr/local/apache2/htdocs
[root@test htdocs]# vi test.php
<?php
phpinfo( );
?>
# 以我的测试主机为例，我的测试主机 IP 为 192.168.1.2 ，所以随便以一部可以联机的 PC，
# 在网址列输入 http://192.168.1.2/test.php
# 或者直接在本机的 X-Window 上面输入 http://localhost/test.php 亦可！
```

27.

在上面的最后一个步骤中，如果你的浏览器有出现类似底下的画面，那么就是编译成功啦！ ^_^，而且，主要的 PHP 组态档案会是在 /usr/local/php4/php.ini 这个档案喔！



System	Linux server.cluster 2.4.20-8smp #1 SMP Thu Mar 13 16:43:01 EST 2003 i686
Build Date	Sep 10 2003 17:46:04
Configure Command	'./configure' '--prefix=/usr/local/php4' '--with-apxs2=/usr/local/apache2/bin/apxs mysql=/usr/local/mysql' '--with-config-file-path=/usr/local/php4'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/php4/php.ini
PHP API	20020918
PHP Extension	20020429
Zend Extension	20021010
Debug Build	no
Thread Safety	disabled
Registered PHP Streams	php, http, ftp

哈哈！费了九牛二虎之力，终于将 LAMP 以 Tarball 安装完毕，并且也测试 OK 啰！

升级与安装方式的选择建议：

推荐使用 RPM 的原因：
一般而言，如果没有特别的需求的话，那么使用 RPM 通常已经可以满足大家的需求了！并且在安装上面确实比较容易！此外，未来在升级方面，可以由各家的 Linux distributions 来提供适当的升级版套件，因此，升级上面也是绝对没有问题的！所以这里特别推荐使用 RPM 的方式来进行您的 LAMP 的架设！不但方便简单，而且基础功能都已经含有了！不需要再考虑有的没的咚咚！

有些时候， Tarball 也是不得已的：
不过，有些时候也是不得不升级的呐！或许是因为 distribution 厂商提供的版本速度更新太慢，或者是网络上的 Bug 问题太多，或者是你需要新版本的某些功能时，那么您只好自行以 Tarball 的方式来安装了！而这里需要特别强调的是，如果您使用 Tarball 来安装 LAMP 时，需要整体考虑喔！因为 PHP 有使用到 Apache 2.xx 版本的某些功能，所以重新编译 Apache 时，需不需要重新编译 PHP 呢？绝大部分的情况之下，由于 Apache 支持动态函式库，因此或许不需要重新的编译 PHP ，不过，还是需要注意的！并且，使用 Tarball 的时候，最好全部这三个咚咚都使用 Tarball ，不要一个 RPM 一些 Tarball 的，容易错乱啦！

主机设定：

终于来到主机设定的地方了！在底下的设定里面，我们讲的是大方向喔！还有很多的设定可能要大家有兴趣的多多自行试看看啰！此外，由于每个版本的 Linux distribution 都会将 Apache 作一些调整，因此，不见得每个 Linux distribution 的设定档案都会在同一个地方！举个例子来说， Red Hat 6.x 版本以前的 Apache 会用到三个设定档案，但是目前新推出的几个 Red Hat 版本中，只要 httpd.conf 这个档案设定 OK 就好了！另外， Mandrake 使用 RPM 安装后，httpd.conf 竟然不是主要设定档！他的 Apache 已经改到 commonhttpd.conf 啰！还真是奇怪～无论如何，如果您使用的是 Tarball 安装 Apache 2.x 的话，那么设定文件与套件的目录架构，应该就会差不多了！而且，请善用 locate 与 find 来搜寻 Apache 与 PHP 还有 MySQL 的设定档喔！因为他们的设定档名称应该是不变的！即使 Mandrake 修改了部分内容到 commondhttpd.conf ，但同目录之下，仍然有 httpd.conf ㄋㄟ！此外，底下我们的设定主要以 Tarball 安装的 Apache 2.xx 版本来谈论。基本上，不论是 1.3.xx 还是 2.xx 版本，主要设定的内容都大同小异啦！看看就能够明了了！

LAMP 的套件结构与主机规划

因为 LAMP 主要有三个基本套件，分别是 Apache/MySQL/PHP，那么这三个套件的设定档与主要执行档在哪里呢？

关于 Apache 的路径问题：
我们在浏览网站的时候，总会进入该网站的『首页』吧，那个首页放置的目录就是所谓的『主网页目录』了。主网页目录我们可以在 httpd.conf 里面来设定，不过还是有套件的默认值的啦！

底下我们就来谈一谈吧！ ^_^

- 主要设定档 httpd.conf ：Apache 的主要设定档之档名为 httpd.conf ，因为不同的安装方式与不同的 Linux distribution 下，这个档案放置的地方会不相同！举例来说，Red Hat 9 的预设路径在 /etc/httpd/conf/httpd.conf ，而我们以 Tarball 安装之后，该档案则是在 /usr/local/apache2/conf/httpd.conf 啰！请依照您的安装情况来判断这个档案的所在喔！

- 主网页目录：主网页的目录预设在 /var/www/html 这个目录，而如果是以 Tarball 安装的话，则预设在 /usr/local/apache2/htdocs 这个目录当中呢！不过，这个主网页的目录是依据 httpd.conf 里面的设定而改变的！

- 登录档：一般来说，登录档我们会希望放置到 /var/log/httpd 里面呢！统一管理比较好一点了！

重点其实就是 httpd.conf 所在的目录啦！因为我们在 Apache 里面进行的各种设定都是在该档案里面进行的呐！ ^_^

关于 MySQL 的路径问题：
MySQL 是数据库软件呀！所以最重要的当然就是他的数据库放置的目录啰！呵呵！答对了！那么这个数据库放在哪里呢？预设是在/var/lib/mysql 里面的！所以如果你有使用 MySQL 做为你的网页或者是论坛的话，那么请将这个目录的咚咚备份下来吧！很重要的喔！

关于 PHP 的模块问题：
一般而言，我们大多以『动态函式库』的方式来进行 PHP 的模块编译的！最大的优点就是可以在升级的时候，只要直接升级 PHP 即可！与 Apache 的相关性就比较小一点，因此，在升级某个套件的时候，不需要全部的套件都一起升级的啦！这样真的是比较简单喔！

好了，那么如何规划我们的 Apache 主机呢？如果您的主机想要提供使用者来设计个人的首页，那么自然就得开放使用者具有家目录啰！而为了安全起见，当然家目录 (/home) 独立一个 partition 是比较好的！并且，由于未来可能会针对不同的使用者进行磁盘配额 (quota) 的限制，所以， /home 真的可以考虑独立一个 partition 的啦！此外，如果您的 apache 未来连接的 Client 数量挺大的时候，呵呵！硬盘与内存需要大而且速度快一点的才好喔！

基本要求：

要让您的 Apache 就是 WWW 能够在 Internet 上面被搜寻到，您的主机最好要有个 domain name ，亦即是有个独一无二的名字啦！如此一来，在 Internet 上面，大家比较容易找的到您的主机！并且，如果您主机的内容已经大致上完备的话，那么可以到各大搜寻引擎去注册(免费)一下，例如奇摩雅虎、蕃薯藤、新浪网等等去注册一下！这样就可以让大家搜寻到你的网站啰！所以，你需要：

4. 连上 Internet ：这不是啰唆的很吗？没错啦！但还是要提醒，既然要对 Internet 公开，第一件事就是要连上 Internet 呐！

5. 申请领域名称 (domain name)：其实应该称为 FQDN 比较好的！ ^_^ ！如果你使用的联机方式取得的 IP 是非固定的，亦即俗称的浮动式 IP ，那么可以选择类似 www.adsldns.org 之类的免费动态 IP 之 DNS 系统！还颇好用的！详情请参考：申请合法的主机名称一文。那么如果想要自行申请 domain name 的话，那么就可以参考 DNS 的设定啦！但请注意，该服务『最好有固定 IP 』者才适用喔！总而言之，就是当 Client 在找 domain name 就需要可以找到 IP 啦！但要特别留意，如果您希望未来架设虚拟主机的话，那么就务必要有授权的 DNS 啰！

6. 安装好 LAMP ：又是废话！ @_@ 反正一定要好好的安装就是了。这里又要特别强调，除非你有很好的逻辑观念，那么你同时安装 RPM 与 Tarball 还没有关系 (同时存在两套 LAMP) ，反正可以读到正确的即可！但是，如果你没有良好的逻辑概念，那么，如果要安装 RPM ，请将 Tarball 的移除！如果要安装 Tarball ，请将 RPM 移除！当然，数据库与网页的数据请记得备份！这样会比较简单啦，而且不容易发生错误！

7. 具备简易的 HTML 语法概念：这个鸟哥就没有办法教大家了！因为不在 Linux 范围内～有兴趣的可以到一个十分棒的网页建置教学网站：网站建置百宝箱 (http://dob.tnc.edu.tw/index.php)！因为具备简易的 HTML 观念后，才比较容易理解整个 Apache 运作的数据之传递的状况！

好了！不啰唆，立刻来实作看看先！

最简易 Apache 设定

为什么挂了个『最简易』呢？呵呵！因为那就是没有任何麻烦的设定嘛！基本上，如果你按照鸟哥上面 Tarball 的方法安装好之后，其实立刻就有主机的首页了！当然，使用 RPM 也有首页啦！因此，用最原始的设定其实就可以进行主机的首页浏览了！需要特别注意的是，第一次进入到 Apache 之后，你会发现有个 documentation ，在那个超级链接当中，对于 Apache 有『相当完整的介绍！』如果有任何问题，可以到该连结里面去搜寻！绝对可以找到您要的信息喔！底下我们就以 Apache 2.xx 版本的设定档案来介绍，如果您是 1.3.xx 版本，请自行参考底下的设定来解决您的 httpd.conf ！设定方面差不多啦！ ^_^

此外，需要注意的是基本的环境设定方法为：

```
<设定项目>
    ......
    ......
</设定项目>
```

例如：

```
<Directory>
    Options Indexs
</Directory>
```

几乎都是这样的设定方式喔！请注意一下即可！特别留意的是，如果你有额外的设定时，不能随便在 httpd.conf 里头找地方写入！否则如果刚好写在 <Directory>...</Directory> 里面，呼呼！那么就会发生错误啦！需要前前后后的找一找喔！

- 基础环境设定：

```
[root@test root]# cd /usr/local/apache2/conf
[root@test root]# vi httpd.conf

ServerRoot "/usr/local/apache2"
# 最上层的 Apache 目录！我们安装的时候，以这个目录来安装的，他就是 ServerRoot 啰！
# 其实，也就是说，如果底下以『相对路径』的方式写的，那么就是相对于这个路径！
# 当然，写绝对路径就没有任何影响啦！

PidFile logs/httpd.pid
# 不要跟我说不晓得 PID 是什么？查看一下 Linux 基础的资源管理去！
# 这个项目在设定 Apache 的 PID 记录文件！可以用在重新读取设定文件等等的功能！
# 如上面所言，因为写了相对路径，所以实际的目录为 /usr/local/apache2/logs/httpd.pid
# 通常我也喜欢将他移动到 /var/log/httpd 底下去，统一管理较方便！
# 我喜欢将这行改变为 PidFile /var/log/httpd/httpd.pid

Timeout 300
# 这是用来设定连接到你这部主机的客户端，当超过 300 秒客户端还没有
# 办法连上你的主机时，就予以断线处理！

KeepAlive On
# 是否允许持续性联机，亦即一个联机有多个要求！这里通常设定为 On 比较好，
# 就鸟哥的经验来看，设定为 Off 似乎会产生很多 Time_Wait 的封包！粉怪！

MaxKeepAliveRequests 100
# 在持续性的联机当中，最多允许的联机数目！如果不要限制，可以设定为 0 ，
# 当然，官方网站上面说，要有较佳的效能，最好设定大一点，所以我都将他改为 200 以上。

KeepAliveTimeout 15
# 同一个联机的 Client 下次的需求没有在 15 秒内送出，那么该联机会被视同断线喔！
```

```
<IfModule prefork.c>
StartServers         5
MinSpareServers      5
MaxSpareServers     10
MaxClients         150
MaxRequestsPerChild  0
</IfModule>
<IfModule worker.c>
StartServers         2
MaxClients         150
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild  0
</IfModule>
# 这两段主要是与系统的效能较有关系！如果不需要效能设定的话，那么使用默认值就很够了！
# 1. MinSpareServers 与 MaxSpareServers 是开启 httpd 服务数目的地方，当你执行
#    /usr/local/apache/bin/apachectl start 之后， 在 shell 下执行
#    ps -aux|grep http 就可以看到 http 的数量， 通常这与你的 RAM 有关。
#    如果是小站的话，可以设小一点， 例如最小设 3 最大设 5 即可！
# 2. 而 StartServers 则设与 Min 相同即可！
# 3. 至于 MaxClients 则可以设小一点，因为设定太大很耗系统资源，
#    而太小则无法让很多人连上来！所以可以设成例如 100

Listen 80
# 设定监听的 port ，如果你要更改 WWW 的 port number ，可以在这里修改，例如 8080

User nobody
Group nogroup
# 这个是设定 apache 所产生的，就是刚刚我们上头所设定的 MinSpareServers ，之后会产生
# 一些 process ，那么这些 processes 的拥有者与拥有群组 ( owerner & group ) 是谁！
# 这个与未来的『 PID 权限及 Linux 权限设定』有关！通常如果是 RPM 安装的话，
# 大致上都会是 apache ，而如果是 Tarball 安装，通常是 nobody 与 nogroup 吧！
# 是否有该 user 还要查看 /etc/passwd 及 /etc/group 喔！不能设错！否则无法启动 apache

ServerAdmin root@localhost
# 这个是设定你的机器的 httpd 管理员账号！设成你的 e-mail 吧！例如我都设定为：
# 我的机器上：ServerAdmin  test@localhost

#ServerName new.host.name:80
# 设定主机名称的地方，若有需要的话才设定，否则可以将他 mark 掉也没关系！
# 但如果你需要让 Apache 自动帮你将其它名称连过来的主机名称修改时(下一个设定)，
# 那么这里就需要填写啰！此外的此外，需要了解的是，有时，例如 Openlinux server 3.1.1
# 如果没有设定这个的话，那么你的 WWW 将无法启动！另外，如果设定错误，同样无法启用！
```

```
# 因此，没有特别要求的话，那么这个就暂时不要设定吧！免得自找麻烦～

UseCanonicalName Off
# 主机的别名啦！例如你的主机有三个名称时，那么这个 Off 的设定，会让 Client 端
# 可以分别使用三个名称显示在他们的浏览器上面，如果是 On 的话，那么将以上面的
# ServerName 内容来显示在他们的浏览器上面，而不是原来他们写的主机名称喔！
# 说是这样说，但是我试不出来这个功能～～ @_@

AddDefaultCharset ISO-8859-1
LanguagePriority en da nl et fr de el it ja ko no pl pt pt-br ltz ca es sv tw
# Apache 的预设显示语言编码！请特别留意，因为这里的设定并不适合台湾的繁体中文，
# 呵呵！所以底下请『务必』修改成这样！否则您的网页总是无法显示中文喔！
# 这两行大概在 httpd.conf 的 750 ~ 800 行之间！请以搜寻的方式找一下！
AddDefaultCharset Big5
LanguagePriority tw en da nl et fr de el it ja ko no pl pt pt-br ltz ca es sv

HostnameLookups Off
# 在记录档案的时候，登录档的内容，来捉取我们数据的主机是以 IP 还是主机名称来显示？
# 当然是 IP 来显示比较快喔！所以，这里通常设定为 Off ，不需要转译 IP 成为主机名！
```

o

o 目录路径设定：

```
[root@test root]# cd /usr/local/apache2/conf
[root@test root]# vi httpd.conf

DocumentRoot "/usr/local/apache2/htdocs"
# 将 /usr/local/apache2/htdocs 设定为 Apache 的根目录！
# 这个就是主机的主网页啦！你可以将他移到任何你高兴的地方！
# 不过，比较重要的限制是，最好这个目录底下不要包含重要的信息，例如你不要将根目录 /
# 设定为这个 DocumentRoot 吧！ ^_^ 否则你的主机下的任何数据，不就任何人都可以使用
# 浏览器来查看？岂不是很危险！

<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
# 这个是设定根目录，亦即是 /usr/local/apache2/htdocs 这咚咚的属性啦！
# 使用 Options 来设定相关属性，相关的属性还有底下几个ㄋㄟ！
ExecCGI     ：使该目录底下的 CGI 具有可以执行的能力！重要项目！如果您要您的
              某个目录可以执行 CGI 的程序时，那么请将该目录多加 ExecCGI 这个属性！
FollowSymLinks  ：让您的 link 的目录或档案，虽然在其它的目录下，仍然可以连接出去！
                  举个例子来说，目前我的主页是 /usr/local/apache2/htdocs ，但我想
```

```
                 连接到 /home/vbird/testing 底下，然而我又不想多加一个 directory
                 的设定值在 httpd.conf 内，那么我可以在 /usr/local/apache2/htdocs
                 使用 ln 连结一个名为 vbird 使他指向 /home/vbird/testing ，那么
                 当我网址输入 http://localhost/vbird 时，就可以到
                 /home/vbird/testing 了！
                 如果没有设定这个属性，那么就无法连接出去喔！
Includes    ：在 Server 端的工作可进行！
Indexes    ：如果在该目录底下找不到 index.html 时，就显示整个目录下的文件名称！
             粉危险吧！ ^_^ 所以啰，尽量不要包含 Indexes 这个项目啦！
MultiViews  ：这个东西有点类似多国语言支持啦！你可在同一目录下的同一个档案，
             编写多个不同语言的档案，并且以一个 *.var 的档案来规范不同编码！
             有兴趣的话，请自行参考自己的 /usr/local/apache2/htdpc/index.html.var
All ：全部的属性都启动啦！但是不包含 MultiViews ！

<Directory "/usr/local/apache2/htdocs">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
# 1. 这段主要在设定 / 的属性啦！可以看到预设的情况中有 indexes 喔！所以赶紧将他拿掉
# 2. 至于 AllowOverride 主要与认证网页的设定有关啦！亦即 .htaccess 档案！意思是说，
#    httpd.conf 在与 .htaccess 相同目录下的设定中，哪些设定会用来取代 .htaccess 的
#    设定内容？当然啦， .htaccess 这个档案设定较为灵活，所以当然设定为 None 啰！
#    仍然是以 .htaccess 的设定为主啦！
# 3. Order allow,deny 注意喔，那个 allow,deny 中间是以 , 隔开，并没有空格！不要
#    设定错了！这个东西与底下的 Allow 在设定『权限』的啦！我们后面会再详谈！

UserDir public_html
# 这玩意儿在设定个人家目录下的首页在何处啦！这里预设是 public_html ，举例来说，
# 我的家目录是 /home/test ，那么这个使用者的首页目录在 /home/test/public_html ！
# 当然，这个目录是可变的！就看你要怎么设定啰！ ^_^
# 例如很多人都喜欢将这个目录设定为 www ㄋㄟ！呵呵！也顺道去修改一下 /etc/skel 的内容

DirectoryIndex index.html index.html.var
# 这个就是当我们输入 http://192.168.1.2 时，那么 Apache 将会去搜寻该目录底下的文件名！# 预设只有
个，太少了！如果我们使用 php 之类的，哇！那可就糗了！所以，这里可以改成
DirectoryIndex index.html index.htm index.php index.cgi index.php3 index.html.var
# 如果还有喜欢的预设档名，将他加进去吧！这就是首页的网页名称啰！ ^_^

Alias /icons/ "/usr/local/apache2/icons/"
<Directory "/usr/local/apache2/icons">
    Options Indexes MultiViews
    AllowOverride None
```

```
    Order allow,deny
    Allow from all
</Directory>
# Aliase 之设定主要也是在简化一些繁复的连结内容啦！举上面的例子来说，我们的 Apache
# 根目录在 /usr/local/apache2/htdocs 里面，那么输入网址 http://localhost 则到该目录
# 在上面的设定中，则输入 http://localhost/icons 会跑到 /usr/local/apache2/icons
# 的意思！如此一来，设定上较为简便了！

Alias /manual "/usr/local/apache2/manual"
<Directory "/usr/local/apache2/manual">
    Options Indexes FollowSymLinks MultiViews IncludesNoExec
    AddOutputFilter Includes html
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
# 这个与上面的设定相同！所以，未来只要输入 http://localhost/manual 即可到达自己的
# 主机上面的说明文件喔！很简易吧！ ^_^

ScriptAlias /cgi-bin/ "/usr/local/apache2/cgi-bin/"
<Directory "/usr/local/apache2/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
# ScriptAlias 这部份则大概专属于 CGI 之类的可执行程序吧！呵呵！要了解一下，
# 我们预设都是将 http://localhost/cgi-bin 设定为可执行的 CGI 程序放置区！
# 那么 CGI 程序要开放的权限有哪些！呵呵！就上面设定的啰！
```

-
- 启动 PHP 与 CGI 相关模块，及预设的登录档修订！：

```
[root@test root]# cd /usr/local/apache2/conf
[root@test root]# vi httpd.conf

LoadModule php4_module        modules/libphp4.so
AddType application/x-httpd-php .php
AddHandler cgi-script .cgi .pl
# 至少这三行必须要启动喔！尤其是第三行，通常预设都是关闭的！所以，你必须要将 # 移除
# 否则你将无法执行 CGI 的程序ㄋㄟ～

ErrorLog logs/error_log
```

```
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
CustomLog logs/access_log combined
# 上面全部都跟 log 有关，因为鸟哥真的比较喜欢将 log file 放在一起处理，所以我都会
# 这样改：
ErrorLog /var/log/httpd/error_log
CustomLog /var/log/httpd/access_log combined
# 只改这两行，其它的保留默认值咯！ ^_^
```

o

这样就给他设定完成了啦！粉不错吧！ ^_^接下来，介绍一下怎么启动吧！

启动 httpd (如何关闭 https)

要启动 WWW 实在是太简单啦！直接给他启动即可！如果是以 RPM 安装的，那么启动档案预设在 /etc/rc.d/init.d/httpd 这个档案，所以，你可以这样启动：

```
[root@test root]# /etc/rc.d/init.d/httpd start (启动)
[root@test root]# /etc/rc.d/init.d/httpd stop  (关闭)
```

至于 Tarball 则是以 apachectl 来作的：

```
[root@test root]# /usr/local/apache2/bin/apachectl start (启动)
[root@test root]# /usr/local/apache2/bin/apachectl stop  (关闭)
```

无论如何，启动之后请立即察看一下 port 是否已经开启了？！

```
[root@test root]# netstat -tuln | grep ':80'
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
```

并且需要在登录档当中发现底下这一行才算成功喔！

```
[root@test root]# vi /var/log/httpd/error_log
[Thu Dec 26 20:16:53 2002] [notice] Apache/2.0.43 (Unix) PHP/4.2.3 configured -- resuming normal operations
```

如何关闭 https ：

此外，如果您比较细心留意的话，会发现 Red Hat 9 的 httpd.conf 里面有这一行：『Include conf.d/*.conf』(大约在 162 行的地方)，这表示：『 Apache 在启动的时候，除了读取原来的 httpd.conf 这个设定档之外，也会读取 /etc/httpd/conf.d/*.conf 的所有档案来规范 Apache 的启动状态。』这样的设定无疑是比较灵活的，可惜的是，对于一些初接触 Apache 的朋友来说，可能会觉得有点怕怕的～ 好了，那么我们来看一下， Red Hat 9 里面的 /etc/httpd/conf.d/ 有什么数据呢？

```
[root@test root]# ll /etc/httpd/conf.d/*.conf
-rw-r--r--    1 root     root    814 Feb 10  2003 /etc/httpd/conf.d/perl.conf
-rw-r--r--    1 root     root    459 Jun 30 04:35 /etc/httpd/conf.d/php.conf
-rw-r--r--    1 root     root   1276 Feb 20  2003 /etc/httpd/conf.d/python.conf
-rw-r--r--    1 root     root  11140 Jul 31 23:40 /etc/httpd/conf.d/ssl.conf
```

在上面的档案当中，较为有趣的是 ssl.conf 那个档案，那个咚咚就是我们前面提到的以 SSL 加密的设定值！这个设定值会让我们的系统多了一个 https (port 443) 在监听网络服务呢！有需要这东西吗？当然不需要了！因为我们又不需要使用信用卡数据说～好了，那么如何将 Red Hat 9 的 https 取消呢？呵呵！将 ssl.conf 更改檔名即可！不要使用 .conf 的附檔名就行了！例如底下的方式：

```
[root@test root]# cd /etc/httpd/conf.d
[root@test conf.d]# mv ssl.conf ssl.conf.bak
[root@test conf.d]# /etc/rc.d/init.d/httpd restart
```

这样就可以关闭 https 啰！

测试结果：
相信您应该都会自行测试您的 WWW 是否正常了吧？！没错！ Apache 是否正常，直接在远程的主机，或者是近端的本机上面，以浏览器直接输入你的 IP 来试看看即可知道！而测试 PHP 呢？就用上面我们提到的那支小小的 PHP 程序来进行 PHP 的内容显示，成功就是 OK 啦！ MySQL 呢？直接给他 mysql -u username -p 再输入密码，就可以知道有没有设定成功啦！整个的设定真的是粉简单的啦！通常，如果上面的测试没有成功的话，最大的可能为：

- 网络问题：虽然在本机上没有问题，但不代表网络一定是通的！请确认一下网络状态！例如 Route table, 拨接情况等等；
- 网页问题：例如鸟哥第一次以 Tarball 安装好之后，竟然发现无法显示主机的首页！后来才发现是主机的首页设定错误！导致找不到网页，这个时候，请特别留意浏览器上面的显示讯息，里面包含了无法连进来主机的问题！请提供这样的讯息到讨论区，大家才知道问题出在哪里呐！
- 权限问题：例如你刚刚在上面的 user 设定为 nobody 了，但偏偏要被浏览的目录权限为 750 ，自然就无法让人家联机进去啦！
- 问题的解决之道：如果还是没有办法连接上来你的 Linux Apache 主机，那么请：

1. 察看 /var/log/httpd/error_log 这个档案吧！他应该可以告诉你很多的信息喔！
2. 另外，也要仔细的察看一下你的浏览器上面显示的信息，这样才能够知道问题出在哪里！ ^_^
3. 另一个可能则是防火墙啦！察看一下 iptables 的讯息！

用户的个人网页启动

呵呵！再来则是个人用户啦！如果是个人用户要自己的首页时，要怎么办呢？刚刚我们不是提到了 httpd.conf 中有一项关于个人首页的设定，通常如果你不另行修改 httpd.conf 档案的话，他的默认值都是『 public_html 』这个设定，好了，那要如何设定个人网页呢？假设以 test 这个账号为例，我们可以这样进行：

```
[test@test test]$ cd  # 回到自己的家目录
[test@test test]$ mkdir public_html
[test@test test]$ chmod 755 public_html
[test@test test]$ chmod 755 /home/test
# 在你的客户端家目录中建立了一个 public_html 的目录，
# 并将此目录的权限改成可以让其它人观看，注意喔， apache
# 预设是 public_html ，但是如果你在 httpd.conf 这个档案中
# 改变了目录名称，则必须作适当的修正喔！
```

然后在你的目录中，亦即 /home/test/public_html 当中，建立一个档名为 index.html 的 HTML 档案，然后在 IE 的网址列打入

http://你的网站名称/~test/

则 apache 会自动将 IE 的讯息传到 /home/test/public_html 这个目录中，并搜寻文件名为 index.html 或 index.htm 或 index.php 的檔名！所以说， index.html 是 apache 第一个找寻的档名喔！这就是你的首页啦！
不过，如果您觉得这样实在很讨厌，怎么需要多加一个毛毛虫的符号『 ~ 』呢！真烦，可不可以只使用：

http://你的网站名称/test

就好啦！？当然没问题啦！但是就需要动一点手脚了！最常见的有两种方式：

- 建立连结档：
  这是最简单的方法啦，还记得我们的主页有 Options FollowSymLinks 吧？！呵呵！所以利用 ln 就可以达到我们的需求了！首先，需要找到首页，然后在首页底下输入连结的方法：

```
[root@test root]# cd /usr/local/apache2/htdocs
[root@test htdocs]# ln -s /home/test/public_html test
```

- 这样立刻生效啦！不需要重新修改喔！厉害吧！！ ^_^
- 建立 Alias：
  另一种方是就是使用我们刚刚提到的那个 Alias 功能啦！也是粉简单的ㄋㄟ：

```
[root@test root]# vi /usr/local/apache2/conf/httpd.conf
Alias /test/ "/home/test/public_html/"
<Directory "/home/test/public_html">
    Options FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

[root@test root]# /usr/local/apache2/bin/apachectl stop
[root@test root]# /usr/local/apache2/bin/apachectl start
```

- 请注意：
  1. 在 Alias 后面双引号接的目录『务必』于最后面加上一个 / ，亦即 ....public_html/ 才行！若没有加上 / ，则该行字符串会被视为『档案』而非目录！特别留意这一点！
  2. 那个 <Directory ....> 里面的设定中，关于目录 /home/test/public_html 请『务必』帮他加上双引号喔！否则可能会无法成功的启动！！

好啦！这样你的用户之个人网页就 OK 啰！ ^_^

进阶安全设定：

接着下来，我们想要了解一下，那么 Apache 有没有其它额外的设定呢？

CGI 之执行、Index 显示、查无网页显示之设定：

CGI 之执行：
首先要来提到的是 CGI 的执行问题，这也是很多朋友想要提出的问题啦！到底要怎么设定才可以在某些路径里面执行 CGI 的程序呢？而不是使用纯文字将他 show 出来？！难道要执行 CGI 就非得在 /usr/local/apache2/cgi-bin 这个目录下不可吗？当然不是！有很多的方法可以来设定的！举个例子来说，假设今天有个一般身份的使用者 test ，他想要可以执行 CGI ，那么他的家目录在 /home/test/public_html 底下，而他的程序是放在 /home/test/public_html/cgi 这个目录中，那我可以怎么作呢？同样有两种作法：

- 使用 Options 及 ExecCGI
  你可以在 httpd.conf 这个档案中，找个地方加入底下的文字：

```
[root@test root]# vi /usr/local/apache2/conf/httpd.conf

方法一：
# 先确认在 httpd.conf 当中，底下这一行已经将批注拿掉了！
AddHandler cgi-script .cgi
# 请注意，如果是想要让 .pl 的档案可以执行 ( Perl )，那么上面那行要改写成：
AddHandler cgi-script .cgi .pl

# 再加入底下几行：
<Directory "/home/test/public_html/cgi">
    Options ExecCGI
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

方法二：
# 直接加入底下这几行即可！
<Directory "/home/*/public_html/cgi">
    Options ExecCGI
    SetHandler cgi-script
</Directory>

[root@test root]# /usr/local/apache2/bin/apachectl stop
[root@test root]# /usr/local/apache2/bin/apachectl start
```

- 呵呵！这样该目录立刻就可以执行 CGI 的程序啦！当然啦，那个 cgi 的档案，权限当中也必须要有『可执行，x 』的权限喔!而,如果你要执行某个 cgi 程序,例如 index.cgi 好了，那么就需要填入：http://your.server.name/~test/cgi/index.cgi 啰!

- 使用 ScriptAlias 功能：
  还有另外一个功能也可以达到同样的效果，那就是使用 ScriptAlias 喔！你可以在 httpd.conf 这个档案中，找个地方加入底下的文字：

```
[root@test root]# vi /usr/local/apache2/conf/httpd.conf

# 先确认在 httpd.conf 当中，底下这一行已经将批注拿掉了！
AddHandler cgi-script .cgi

# 再加入底下这一行：
```

```
ScriptAlias /testcgi/ "/home/test/public_html/cgi/"

[root@test root]# /usr/local/apache2/bin/apachectl stop
[root@test root]# /usr/local/apache2/bin/apachectl start
```

- 呵呵！这样也是可以啦！假设你要执行的档案仍然是 index.cgi ，那么，你的执行网址会变成： http://your.server.name/testcgi/index.cgi ！就是这点不同啰！其它的都一样啦！

Index 显示：
刚刚我们前面不是提到 Options 关于 Indexes 的说明吗？对啦！由于 Indexes 加入的时候，找不到 index.html 时，会将该目录下的所有档案以类似 FTP 的画面秀出来，所以很危险，因此，可以的话，尽量不要在 Options 后面加 Indexes 喔！记得将他拿掉！会比较稍微安全一些！
查无网页：
那么好啦，既然不要秀出没有 Index.html 的目录下的所有档案，有没有什么替代的方法，可以让使用者知道该目录下没有这个档案，并提供他另一个网页的连结呢？再举个另外的例子，如果你的网页经过了大量的改版，所以某些原来的档案已经不见了！但是你有很多的老网友，那么他们连过来的时候，将会发生很多的错误，怎么办？

没关系，还记得当你查不到网页的时候， IE 或者是其它的搜寻引擎，会自动的跑出一个窗口，警告说，查无该网页，并提供另一个可能的连结给您是吧？！同样的，我们也可以利用这样的机制，自己建立一个额外的连结数据来给大家使用！如此一来，嘿嘿！找不到网页也没关系，我们已经提供了其它的连结，这样就可以很轻易的引导使用者到适当的目录去浏览啦

- 设定 httpd.conf
  又要设定这个 httpd.conf 档案啦！找到底下的地方，并且将他开启吧：

```
[root@test root]# vi /usr/local/apache2/conf/httpd.conf

# 大概在 880 行左右，可以找到下面这一行：
ErrorDocument 404 /missing.html
# 将前面的 # 拿掉喔！由于我们的 code 有：
100-199：一些基本的讯息
200-299：要求成功的达成
300-399：Client 的需求需要其它额外的动作，例如 redirected 等等
400-499：Client 的要求没有办法完成(例如找不到网页)
500-599：主机的设定错误问题
# 所以，上面的 ErrorDocument 404 就是在无法查询到网页的代号之一啦！
# 而接在这个项目后面的就是网页的名称咯！因为在 / ，所以可以了解他的档案全名为：
# /usr/local/apache2/htdocs/missing.html 还记得 / 在 Apache 就是 DocumentRoot 吧！

[root@test root]# /usr/local/apache2/bin/apachectl stop
```

```
[root@test root]# /usr/local/apache2/bin/apachectl start
```

-
- 编辑 /usr/local/apache2/htdocs/missing.html
  再来则是要编辑这个档案啰！因为只要没有找到对应的网页时，这个网页就会被显示在 Client 端的浏览器上面，喝！你可以这样编啦：

```
[root@test root]# vi /usr/local/apache2/htdocs/missing.html
<html>
<center>
Missing HTML Web Page<br><br>
I can't find any web page for you, <br>
Please contact with me root@localhost, <br>
or press <a href="http://192.168.1.2/manual/">here</a> to see the manual of Apache 2.xx
</center>
</html>
```

- 打上面那个 192.168.1.2 改成你的 IP ，这样就可以啦！注：这个版本很是奇怪！同样的方法，我在其它的所有 distribution 上面都可以成功，唯独以 Mandrake 9.0 即使用 Tarball 方式安装的此一方法，却无法成功！然而若使用 Apache 所预设提供的功能，那么就又可以成功！还真的是很奇怪.....

抵挡 IP 与限制使用者动作的设定(allow, deny, limit)

任何的 Service 都一样，应该都会有『限制联机登入者』的功能！当然， Apache 也不例外！他提供了抵挡 IP 或者是限制使用者进行某些工作的方法！那就是 Allow 与 Deny 的功能，另外，亦有 Limit 的功能可以来使用！

- 抵挡某些 IP 或 domain 来源：
  很多时候我们发现有某些来源的使用者似乎没有很遵守我们网站的约定，可能会砍站啦，或者是可能从事一些让管理员很生气的勾当！当然，你可以使用 iptables 等之类的防火墙功能来挡掉他，不过，我们也可以额外的再以 Apache 来挡他喔！还记得我们在前面的 <Directory> 里面的设定时，常常会看到 order allow,deny 那一行吧！呵呵！那就是我们可以针对的某些设定啦！举例来说，假如你知道 192.168.1.100 这个 IP 的使用者不乖以及 testing.idv.tw 这个 domain 的来源不被信任，你就不让他进入某个目录内，例如 /home/test/public_html/cgi，那么你可以这样做：

```
[root @test root]# vi /usr/local/apache2/conf/httpd.conf
# 额外再新增几行吧！
<Directory "/home/test/public_html/cgi">
```

```
    Options ExecCGI
    SetHandler cgi-script
    order allow,deny
    deny from 192.168.1.100
    deny from testing.idv.tw
</Directory>
```

- 如此一来，嘿嘿！那个 192.168.1.100 及所有来自 testing.idv.tw 这个领域的主机就无法进入你的 /home/test/public_html/cgi 这个目录啰！不过，其它的目录还是可以进入喔！请留意这个现象！所以，如果要将他全部的权限都关闭的话，或许 iptables 还是比较好的选择，不过，如果只是在乎某些目录而已，那么这个选项就不错用了！

- 限制某些功能：
  一般来说， Client 端有哪些功能可以动作呢？大概有：GET, POST, OPTIONS, DELETE 等等，由于 POST 与 DELETE 算是比较高权限的功能，如果今天您有一个目录，在这个目录中对于 192.168.1.50 这个 IP 来源你只许他观看，而不许他贴上数据，那么你要怎样设定呢？我们继续以上面的例子来进行说明好了，在 /home/test/public_html/cgi 这个目录中，你要限制让 192.168.100.0/24 这个网段不能『浏览』，至于张贴文章方面，则仅有 192.168.1.50 这个 IP 可以具有这样的的功能，那么你可以这样修改：

```
[root@test root]# vi /usr/local/apache2/conf/httpd.conf
# 额外再新增几行吧！
<Directory "/home/test/public_html/cgi">
    Options ExecCGI
    SetHandler cgi-script
    <Limit GET>
        order allow,deny
        deny from 192.168.100
        allow from all
    </Limit>
    <Limit POST>
        order allow,deny
        allow from 192.168.1.50
        deny from all
    </Limit>
    order allow,deny
    deny from 192.168.1.100
    allow from all
</Directory>
```

- 嘿嘿！如此一来，就可以将我们的网页数据搞的更安全啦！！ ^_^

主机状态说明网页设定

既然已经安装好了 WWW 主机，除了提供服务之外，重要的是要如何维护啰！嘿嘿！那么是否一定要额外安装其它的套件才能知道目前的主机状态呢？当然不需要啦！我们可以透过 Apache 提供的特别功能来查询主机目前的状态！那就是 mod_status 这个模块啰！如何使用呢？由于 mod_status 是预设一定会安装的模块，所以根本不需要担心加载与否的问题，唯一要担心的，则是需要启动这个模块的相关设定而已！同样的，也是编辑 httpd.conf 这个档案即可，可以这样做：

```
[root@test root]# vi /usr/local/apache2/conf/httpd.conf
# 找到底下这几行，并且将他修改一下：
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from 192.168.1.11
</Location>

[root@test root]# /usr/local/apache2/bin/apachectl stop
[root@test root]# /usr/local/apache2/bin/apachectl start
```

在上面的设定中，必须要特别留意那个 Deny 与 Allow 的设定，说明的是，要察看这个讯息的，只有 192.168.1.11 这个 IP 来的要求才提供！否则一切则予以禁止。当然啦，主机的安全是需要维护与保密的，当然不让人家知道是比较好的呢！然后直接在网址上面输入：http://your.server.name/server-status 这样就可以看到你的主机的一些状态啰！有点像底下的样子：

```
Apache Server Status for 192.168.1.2
Server Version: Apache/2.0.43 (Unix) PHP/4.2.3
Server Built: Dec 26 2002 01:59:03

--------------------------------------------------------------------------------

Current Time: Monday, 30-Dec-2002 01:31:28 CST
Restart Time: Monday, 30-Dec-2002 01:31:13 CST
Parent Server Generation: 0
Server uptime: 15 seconds
1 requests currently being processed, 4 idle workers
W____...........................................................
................................................................
................................................................
................................................................

Scoreboard Key:
```

```
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"C" Closing connection, "L" Logging, "G" Gracefully finishing,
"I" Idle cleanup of worker, "." Open slot with no current process

PID Key:


   31027 in state: W ,   31028 in state: _ ,   31029 in state: _
   31030 in state: _ ,   31031 in state: _ ,



--------------------------------------------------------------------------------
To obtain a full report with current status information you need to use the ExtendedStatus On directive.
--------------------------------------------------------------------------------

Apache/2.0.43 Server at 192.168.1.2 Port 80
```

上面说明的，除了显示出主机目前的 IP 与 时间 之外，亦列出 Linux Server 上面，关于 Apache 的使用程序状态！虽然很简单且阳春，不过该有的还都是有了！蛮不错的ㄋㄟ～有空的时候，可以多多的察看一下 Apache 的状态！不过，特别需要注意一下能够使用这个路径的，最好局限在少数人会较为安全喔！

关于权限的意义说明与设定

在网络的服务当中，最麻烦的地方可以说是『权限』的问题了！怎么说呢？刚刚上面提到了提到了 httpd.conf 当中有设定 User 与 Group 这几个咚咚，而这两个属性是针对 Process 来设定的，也就是说，当你启动了 apache 之后，就会有 httpd 这个服务的 Process 在你的 Linux 系统(内存当中)的背景中执行了，而且这个 httpd 属于刚刚我们设定的 nobody 使用者与 nobody 群组(注：刚好 User 与 Group 的名称都叫做 nobody 啦！^_^)！所以啰，当使用 httpd 这个 Process 来进行 Linux 数据的存取时，那么如何设定 Linux 底下最重要的『权限』问题呢？还记得我们在 Linux 主机之资源管理 里面提到的，所谓的 PID 就是 Process 的 ID，当我们触发一个事件时，Linux 就会给这个事件一个 ID 当作执行时候的识别码，例如当 test 这个人下达『 vi /etc/fstab 』的时候，主机就会给这个事件一个 ID ，此外，这个 PID 还会记录其它的权限喔！例如，因为启动 vi 这个事件的使用者是 test ，所以这个 PID 当然就属于 test 啰！所以当 vi 执行期间所开启的所有档案，都需要针对 test 来察看他是否可以具有读、写、执行的权限呢！

这样说您应该有点了解了吧？是的，你在 Client 是否能浏览 Server 所提供的数据，是跟 Server 内的『权限』设定有关呐！举个例子来说，还记得刚刚我们有将 /home/test/public_html 这个目录设定成可浏览吧！而这个目录属于谁呢？没有错，是属于 test 这个人的！那么我们的 apache 所启动的 process 属于谁呢？以上面的例子为例，那就是 nobody 这个人的！那么万一 /home/test 的属性是『drwx------』，请问你，Client 端的使用者能否浏览这个目录呢？呵呵！

由于 Apache 的 nobody 在 /home/test 的权限当中要寻找的是 others 的属性，嗄！怎么是『 --- 』，自然就是『不行浏览！』无论你的 /home/test/public_html 权限开放的有多大！这样可以了解吗？也就是说，万一上层目录就不许进入了，那么底下的目录当然也就无法被登入啰！

一般而言，使用者容易在个人首页发现『Permission deny』的字眼，最常见的问题就是『Linux 权限设定错误』了！只要针对你的权限去修订，那么差不多就可以解决绝大部分的 Apache 权限问题喔！另外，需要特别留意的是，使用者能否进入一个目录，主要是与可执行与否的权限(也就是 x)有关，因此，如果您要开放静态网页的浏览，那么至少应该要将 /home/test 设定成 drwxr-xr-x ，亦即是 755 才行，不过，如果您要执行的是 PHP 之类的网页程序语言，呵呵！那么您只要设定 drwx--x--x 就可以啰！也就是 711 啦！因为毕竟 Client 只要能够执行 PHP 程序即可，因为结果的显示是在 Client 端呐！与是否能在 Server 端浏览是没有关系的ㄋㄟ～

设定认证网页

是不是有过进入某个网站之后，按下某些连结，竟然出现一个对话框框，告诉你要登入该目录，需要输入账号与密码才能登入？呵呵！那就是所谓的『认证网页』啰！这种认证的模式最起码可以达到最小的保护作用，使你的数据比较保险啦！咦！那么使用 Limit 不就好了？但是 Limit 的规定较为严格，若是改天你去外头的网咖店，然后想要联机进入你的主机作一些事情，如果你设定除了内部 IP 之外，外部就无法以 Web 接口连进来的话，那么不就糗了吗？呵呵！这个时候认证网页可就是你的好帮手啰！另外，目前很多学校老师也会将自己的讲义放在网站上，然后以认证网页的方式提供自己的学生下载使用！所以说，这个也是蛮不错的一个变通方式哩！

那么认证网页怎么搞呀！？说来还真的是很容易ㄋㄟ！

9. 既然我们是『按了某个连结进入某个目录之后，才会出现对话窗口』，那么首先当然就是要有那个设定为认证网页的『目录』啰！请注意，是要目录才行喔！

10. 然后，在对会窗口中，既然我们需要输入 ID 与密码，那么自然就需要密码文件啰！另外，虽然 Apache 有支持 LDAP 及 MySQL 等等的认证机制，不过我们这里并不讨论其它的认证机制，完全使用 Apache 的预设功能而已，所以，底下我们会使用基本（Basic）的认证模式喔！
11. 再来，当然就是到 httpd.conf 档案中去设定我们刚刚建立的那个目录的相关信息啰！
12. 最后，重新启动就 OK 啦！

好了，那么我们来作个例子吧！假设，我要在 http://localhost/protect/ 这个目录下作一个认证数据，在这个目录当中仅有 test.html 这个档案，此外，我要让 test 这个 ID ，密码为 testing 及 qqq 这个账号，密码为 qqqpass 做为登入的账号，那么我该如何设定我的数据呢？一样的，一步一步来进行吧！

13. 制作保护目录：

第一步骤当然是制作保护目录啰！既然这个目录在 http://localhost/protect/ 底下，那么有哪些方式可以达成呢？

- 最简单的方是就是直接在 /usr/local/apache2/htdocs 这个目录下再建立一个名为 protect 的子目录啰！不过，这样似乎太简单了 ^_^
- 再来，如果您的网页支持 FollowSymLinks 这个参数(options)的话，那么在任何一个目录下，只要你在 /usr/local/apache2/htdocs 利用 link (ln) 制作一个连结档，那么也可以达成所要的目的！
- 最后，哈哈！直接使用 Alias 就可以啦！

我们选择使用最简单的目录方式来达成好了：

```
[root@test root]# mkdir -p /usr/local/apache2/htdocs/protect
[root@test root]# cd /usr/local/apache2/htdocs/protect
[root@test root]# echo "This is a protect page" > test.html
```

上面我加一个 -p 的参数可以帮我递归的一直建立好这个目录喔！然后立刻就又做好一个档案ㄋㄟ～

14. 制作密码文件：

制作密码文件只要使用 htpasswd 这个命令就可以啦！他的语法是这样的：

```
htpasswd
语法：
[root@test root]# htpasswd [-c] password_file_name User_name
说明：
-c  ：当后面的 password_file_name 这个密码文件不存在时，那么就建立该档案

范例一：
新建一个档案，并建立 test 这个 ID
[root@test root]# cd /usr/local/apache2
[root test apache2]# htpasswd -c apache.passwd test
New password:
Re-type new password:
Adding password for user test

范例二：
已经存在密码文件了，要新增使用者账号
[root@test apache2]# htpasswd apache.passwd qqq
New password:
Re-type new password:
Adding password for user qqq

[root@test apache2]# more apache.passwd
```

```
test:gPxbCD4QIGFwg
qqq:5qPxrLrxRyRrg
```

15. 没错！这样就 OK 啦！你已经有两个账号(test 与 qqq)在 /usr/local/apache2/apache.passwd 这个档案中啰！不过，这里请注意，由于你的密码文件可以存放在任何地方，但是毕竟这里面有你的重要信息，所以，请特别留意的是，『不要将这个档案放置在浏览器可以浏览到的目录！』举个例子来说，放在 /usr/local/apache2/htdocs 就不是一个明智的选择！因为很可能会被别人浏览到这个档案而失去『认证密码』了，那么岂不是很危险！？所以，尽量给他放在浏览器无法浏览到的地方，例如我们提到的 /usr/local/apache2 这个目录就是一个还不错的目录！

16. 针对保护的目录设定认证的内容：
当然接下来我们要针对那个受保护的目录进行设定啦！就是要又开始搞设定档啰！怎么搞？！就是要加入信息呐！加入哪些信息呢？这当中当然就包含了『密码文件的完整目录与文件名、认证的类型、提示的字符、与允许登入的使用者』如果你看到认证网页的登入窗口时，通常就会看到两个输入 ID 与密码的格子，所以自然要设定密码文件的档案，而你也会看到该窗口上面有一些提示字符告诉你这个网页是干嘛用的！另外，像我们刚刚制作了两个账号，那万一你只允许一个账号登入，另一个账号不许登入，那要如何搞呢？可以这样搞喔：

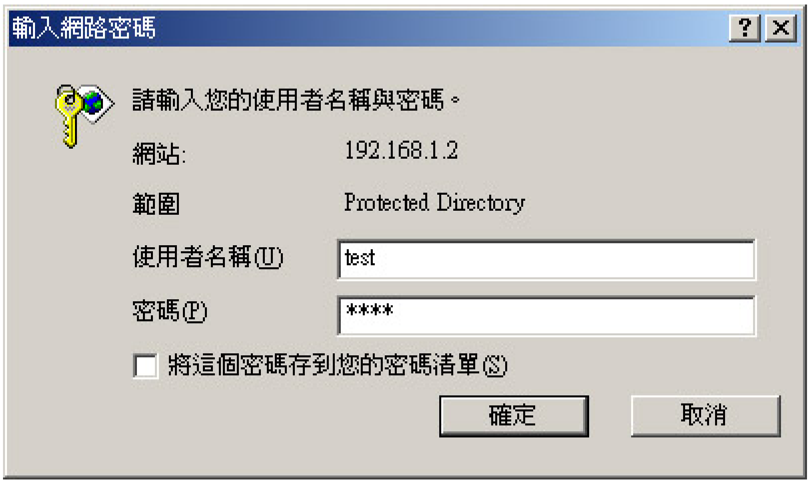
```
[root@test root]# vi /usr/local/apache2/conf/httpd.conf
在这个档案的设定中，请特别留意设定的地方，不要设定错误地方，
一般而言，新的咚咚可以加在最后面一行开始，比较不会搞错地方！
加入底下这几行：
<Directory "/usr/local/apache2/htdocs/protect">
    AuthName "Protected Directory"                      #这个是显示在窗口上面的提示字符
    AuthType Basic                                      #这个则是认证的类型！就选 Basic 即可，Apache 的
    AuthUserFile /usr/local/apache2/apache.passwd       #密码文件放置的地方啦！完整的目录与文件名
    require valid-user                                  #允许的使用者，valid-user 为任何一个在认证档案
    #require user test                                  #若将 # 移除，则表示只有 test 才是可以登入的
</Directory>
```

17. 真的很简单啦！就只要上面的四行设定内容就够了！ AuthName 就是在出现要你输入 ID 与密码的那个提示字符啦！至于谁可以登入呢？以上面的说明为例，当设定为 valid-user 时，表示任何在认证档案中出现的使用者都接受登入，至于如果是 require user test 那一行，则表示可以登入者仅有 test ，如果有两个以上呢？可以写成『 require user test qqq 』的样式喔！

18. 重新启动与测试：
这个不用再讲了吧？！：

```
[root@test root]# /usr/local/apache2/bin/apachectl stop
[root@test root]# /usr/local/apache2/bin/apachectl start
```

19. 测试怎么测试？直接连上网页呐！在网址列输入：
『http://your.host.name.or.IP/protect/test.html』然后应该就会出现：



看到那个 Protected Directory 的字眼了吧！那就是 AuthName 啰！然后使用者名称与密码即是我们刚刚建立的密码文件之内容啦！认证网页设定成功！恭喜喔！

.htaccess 档案与 AllowOverride 设定的用途：

从头到现在有的学的吧？ ^_^ 粉累喔！呵呵！尤其最累的是，如果万一你真的设定好了一个 Apache ，而且也真的很高兴的对外开放了！那么你是否会注意到一件事情，那就是，万一你有 20 的使用者，每个使用者都要求由他设定自己风格的家目录，亦即他们想要自己设定自己的认证网页，想要自行管理自己的 Apache 底下的其它功能，怎么办？由于有关 Apache 的设定就一定需要：

20. 修改 httpd.conf 这个档案；
21. 然后再重新开机，好让设定的结果可以正确的显现出来！

如此一来的话，难道：

- 我就要把 root 的密码给他们吗？好让他们可以自行修改 httpd.conf ？如此一来的话，不就粉危险，而且，难保 A 君的设定被 B 君不小心给修改了～哇！真是伤脑筋～
- 我就要一个一个的帮他们修改吗？谁来就要帮谁改，难道改天我有 200 的使用者时，若一天分配给一个人，我就要改 200 天的 Apache ，岂不讨厌～

这不是很累吗？呵呵！这个时候， .htaccess 的档案就『英雄有用武之地』咯！这个档案的最大功能就是『可以取代 httpd.conf 里面，对于这个 .htacces 所在目录的设定内容！』也就是说，如此一来，每个目录下的 .htaccess 可以让该目录拥有相关的权限、风格等等的设定！如此一来，root 也就不会这么累了，并且，修改完这个档案之后，也不需要重新开机ㄋㄟ！棒吧！ ^_^，不

过，到底 .htaccess 这个档案可以工作的内容有哪些呢？呵呵！这个时候，可就需要 AllowOverride 在 httpd.conf 里面的设定来规定啰！所谓的 AllowOverride 说的是『允许取代某些设定内容』的意思，而这个 AllowOverride 可以取代的数据有：

- Options：就是允许自行设定一些网页参数，例如 Index, ExecCGI....等等
- AuthConfig：就是网页认证的设定内容啦！
- Limit：就是上面我们提到过的关于安全的设定咯！
- All：上面的规定都可以允许 .htaccess 里面来设定
- None：上面的规定都不许由 .htaccess 来规定！

如果你想让 .htaccess 这个档案有最大的取代内容时，可以选择 AllowOverride All，若仅想让这个档案具有取代网页认证的功能，那么就可以使用 AllowOverride AuthConfig 即可！

举个例子来说，刚刚我们设定 /usr/local/apache2/htdocs/protect 为保护目录，需要做的事情最重要的为设定 httpd.conf 这个档案，并且需要设定有的没的一大堆，好了，现在假如我想让所有的 user 都能自行在自己的家目录内进行？怎么利用 .htaccess 这个风格档案呢？这其中当然涉及了(1)root 对于 httpd.conf 的设定以及(2)一般身份使用者对于自己家目录下的设定！好了，现在我们分别以两种身份来设定自己家目录下的咚咚，以 test 这个使用者为例好了，来看看怎么使用认证网页在一般使用者家目录下！

```
        关于 root 的设定项目：

0. 先确认底下这些信息可以在您的 httpd.conf 里面发现：
[root@test root]# cd /usr/local/apache2/conf
[root@test conf]# vi httpd.conf
AccessFileName .htaccess  <==表示你的设定档案的档名！
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
# 上面这个信息主要就是定义出你在某个目录底下有个控制档案，那个档案的档名啦！

1. 建立让每个使用者家目录下都能自行设定 AuthConfig 的规则！
[root@test root]# vi /usr/local/apache2/conf/httpd.conf
# 在一个新的角落(可以在最后一行加入)加入底下的字眼
<Directory "/home/*/public_html/">
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
</Directory>
# 上面的说明是：在每个使用者家目录下，都可以使用 AuthConfig 的设定
# 在 .htaccess 这个档案中！设定完毕之后重新启动 Apache
```

```
3. 重新启动 Apache 啰!
[root@test root]# /usr/local/apache2/bin/apachectl stop
[root@test root]# /usr/local/apache2/bin/apachectl start


        关于一般身份使用者 (test 为例) 的设定项目:

0. 以一般身份使用者登入，或者使用 su 转换 test 的身份
[root@test root]# su test

1. 建立保护目录与网页内容
[test@test test]$ mkdir public_html
# 上面说明的是我要建立 test 底下的一个名为 public_html 的目录!
# 还记得上面提到的个人首页的设定吧! 对啦! 这个 public_html 请依你的规定设定!
# 此外，这个 test 必需已经存在你的 /etc/passwd 当中了!
[test@test test]$ cd public_html
[test@test public_html]$ mkdir protect2; cd protect2
[test@test protect2]$ echo "protect2 web page" > testing.html
# 上面可以建立目录与网页内容喔!

2. 建立 test 自己的密码文件
[test@test test]$ htpasswd -c /home/test/apache.passwd test
New password:
Re-type new password:
Adding password for user test
# 在 /home/test 底下建立密码记录文件，文件名 apache.passwd ,
# 并且建立起一个名为 test 的使用者!

3. 建立 .htaccess 档案的内容!
[test @test protect2]$ vi .htaccess
# 加入底下的字眼:
# This file is used to test the .htacces function
AuthName "Protect test by .htaccess"
Authtype Basic
AuthUserFile /home/test/apache.passwd
require user test
# 不论你信不信，这个时候当你要进入 http://your.host.name/~test/protect2/testing.html
# 就需要有认证密码才行了! 很快乐吧!
```

赶快去测试看看吧！这样对于 Root 来说，是比较轻松一点，对于使用者来说，可以使用的规范也比较多样化！但是，衍生出来的安全问题，可能就需要大家共同的维护了！当然啦，你也可以使用 AllowOverride 来设定更多 .htaccess 可以规范的项目呢！

防火墙

一句老话啦！就是『如果你的 Server 不能动，但是确定设定都没有问题，那么除了持续查询 log file 之外，最大的可能就是防火墙挡住了！』这句话应该没有什么太大的疑问才对～所以呢，万一你的 Apache 不能动，那么请看看你的防火墙设定规则吧！因为防火墙我们已经提过了，请参考『简易防火墙设定』一文，这里不再赘述！

登录档分析与其重要性：

无论怎么说，登录档永远是相当重要的！尤其是在 /var/log/httpd 里面的 error_log 这个档案！因为：

1. 他记录了所有试图进入你主机读取 apache 的网页数据，但是却失败的所有纪录，
2. 此外，如果你的 Apache 设定错误，那么问题也是记录在这个档案中；

而至于其它相关的档案，嘿嘿！那个 /var/log/httpd/access_log 档案也需要注意一下，因为他记录了所有来自 Client 端的 IP 以及其它相关的读取数据之讯息！这个档案对于未来分析你的 Apache (WWW)被读取网页的所有纪录喔！相当的重要的啦！

---

syslog 与 logrotate：

首先针对 Apache 的设定档当中，要注意的就是那个 syslog 与 logrotate 啰！详细的信息可以参考一下『鸟哥的 Linux 私房菜 -- 基础学习篇』之认识登录档。好了，那么你的 syslog 应该要怎么设定呢？因为我们毕竟是使用 tarball 安装的，所以根本就没有所谓的 syslog 的设定，不过，这个不需要担心，这是因为 Apache 里面本来就有 syslog 的设定存在了！所以，这点我们不需要重新来设定啦！只要记得你的登录文件目录是在哪里就可以啦！目前我的登录档与 pid file 是在：

- /var/log/httpd/access_log
- /var/log/httpd/error_log
- /var/log/httpd/pid

然后，我的 logrotate 的纪录目录在 /etc/logrotate.d 里面，因此，我就在里面新增一个档案，称做是 apache，你可以这样做：

```
[root@test root]# cd /etc/logrotate.d
[root@test logrotate.d]# vi apache
# 新加入这几行
/var/log/httpd/access_log /var/log/httpd/error_log {
    rotate 4
```

```
   missingok
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/log/httpd/httpd.pid 2>/dev/null` 2> /dev/null || true
    endscript
    compress
}
```

如果不知道上面的每个数据代表的意义，还是请前往 认识登录档 一文喔！然后赶快试看看是否可以执行 logrotate 呢？

logrotate -f /etc/logrotagte.conf

呵呵！这样就对啦！那么为什么要加入这个 logrotate 呢？这是因为，未来，如果你的 WWW 服务器越来越大时，那么应该 access_log 档案会『很可怕的大！』例如目前本小站的流量每周可以造成我的登录档长大到 400MB 以上～如果不将他 rotate 的话，哈哈！不出几个星期，我的硬盘就爆了～所以， logrotate 是很重要的喔！

Web Analyser

如果我想要知道最近有谁曾经来我的网页上面逛过，以及该使用者使用的是什么样的操作系统呢？该怎么看？！很简单呐！直接跑到 /var/log/httpd/access_log 这个档案里头去瞧一瞧就知道啦！里面的纪录有点像这样：

```
192.168.1.11 - - [27/Dec/2002:00:20:24 +0800] "GET /manual/ HTTP/1.1"  200 7340
  来源 IP               日期与时间                动作与网页          动作代码
```

立刻可以知道在何时，那个 IP 对于本机的动作是什么，以及是否有成功？很清楚对吧！所以我们可以藉由这个档案知道我们的主机被利用的状态！但是，如果像我的主机一样，一下子就有 400 多 MB 的档案，您要怎么分析？看都看不完ㄋㄟ～呵呵这个时候，就需要使用 Shell scripts 的帮助啦！你可以自行写一个适合您自己的分析工作，来进行解析的行为！不过，由于目前网络上已经有很多的好用的 Web 分析的工具啦！所以呢，我们只要进行他的安装即可喔！很方便吧！底下我们介绍几个好用的 Web 分析工具来给大家瞧一瞧！

Webalizer 网页分析工具

- 官方网站：http://www.mrunix.net/webalizer/
- 设定难度：简单，极适合新手架设
- 软件特色：大致上，所有分析的内容他都有了！虽然图表比较没有那么炫...
- 授权模式：GPL

这个是很不错，也且功能也都很完备的一个网页分析软件！不论是在安装与设定上面，

都是粉简单的ㄋㄟ～所以才说他是极适合新手来安装的一个软件呐!此外,由于他是 GPL 授权码的软件，所以呢，嘿嘿！很快乐的下载吧！请赶快到官方网站下载一下啰！

整个安装流程上面很简单，最重要的地方只有在设定的一些小步骤需要留意而已，好了，废话不多说，我们直接来安装与测试一下，就知道他是怎么回事啦！不过，在安装与设定之前，你必须要先知道你系统里面的 log file 在哪里，以及未来要安装在何处？！我的规划是这样的：

- 我的 Apache 登录档案为 /var/log/httpd/access_log 这个档案！
- 预计直接将软件安装在 /usr/local 底下(Webalizer 提供了反安装！)
- 预计将输出的内容传导到 /usr/local/apache2/htdocs/webalizer

好了，开始来下载、安装与设定吧！你可以依照底下的网站来下载，不过不保证该档案会继续存在～你也可以到『http://linux.vbird.org/download/index.php#webalizer』来下载档案！

```
0. 确认一些必须要的绘图相关 RPM 已经安装!
# 因为这个套件需要 gd, zlib 与 png 才行，所以，你需要安装这三个咚咚!
# 在 Mandrake 9.0 当中，你所需要的套件在光盘中的名称为:
libpng3-devel-1.2.4-3mdk
libpng3-1.2.4-3mdk
zlib1-1.1.4-3mdk
zlib1-devel-1.1.4-3mdk
libgd1-1.8.4-6mdk
libgd1-devel-1.8.4-6mdk
# 至于在 Red Hat 7.2 当中，你要的套件名称为:
zlib-1.1.3-25.7
zlib-devel-1.1.3-25.7
libpng-devel-1.0.14-0.7x.3
libpng-1.0.14-0.7x.3
gd-1.8.4-4
gd-devel-1.8.4-4
# 请一定要安装喔! 不然肯定无法安装这套软件的!

1. 下载软件：可以直接到官方网站，或者到我们网站上下载:
  http://linux.vbird.org/download/index.php#webalizer
[root@test root]# wget ftp://ftp.mrunix.net/pub/webalizer/webalizer-2.01-10-src.tgz

2. 安装软件，同样的，到 /usr/local/src 下面解压缩喔!
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/webalizer-2.01-10-src.tgz
[root@test src]# cd webalizer-2.01-10
```

```
[root@test webalizer-2.01-10]# ./configure --prefix=/usr/local  \
> --with-language=chinese
[root@test webalizer-2.01-10]# make
[root@test webalizer-2.01-10]# mkdir -p /usr/local/man/man1
[root@test webalizer-2.01-10]# make install
[root@test webalizer-2.01-10]# mkdir -p /usr/local/apache2/htdocs/webalizer
# 这个时候系统就已经将软件安装在你的系统上啰！你看，很简单吧！ ^_^

3. 设定文件编修：
# 由于 webalizer 的基本设定档在 /etc/ 底下，不过你需要更改名称之后才行动作！
[root@test root]# cd /etc
[root@test etc]# cp webalizer.conf.sample webalizer.conf
[root@test etc]# vi webalizer.conf
# 只要修改底下这几个重要的信息即可：
LogFile      /var/log/httpd/access_log   #这个就是你的 Apache 登录文件完整路径与文件名
LogType      clf                         #选择 log file 的格式，就是 clf 这种啦！
OutputDir    /usr/local/apache2/htdocs/webalizer #当数据处理完毕之后，输出的目录
Incremental  yes                         #当你的 logrotate 不是一个月一次时，必需设定！
HostName     test.vbird.idv.tw           #输出档案的时候，显示在最上方的主机名称
# 其它的则使用默认值就可以啦！很简单的啦！

4. 测试 Run 的结果
[root@test etc]# webalizer
Webalizer V2.01-10 (Linux 2.4.19) Chinese
使用记录文件 /var/log/httpd/access_log (clf)
产生输出于 /usr/local/apache2/htdocs/webalizer
主机名称是 'test.vbird.idv.tw'
历史记录(history file)找不到…
Previous run data not found...
产生报表给十二月 2002
Saving current run data... [01/09/2003 16:57:18]
产生报表给一月 2003
产生汇总报表
储存历史记录信息
2687 记录 in 1.15 秒, 2336/sec
# 瞧！这样就是执行 OK 啦！然后规定一下，每天跑一次ㄋㄟ！

[root@test etc]# vi /etc/crontab
# 加入底下这一行：
20 2 * * *  root /usr/local/bin/webalizer
# 说的是每天的 2:20 执行一次 webalizer 喔!
```

接下来要测试啦，直接给他 http://your.host.name/webalizer 这个目录的最后面那个 webalizer 跟上面你设定的 Output 的目录有关，请依照你的主机刚刚的设定去规定他

吧！结果呢？呵呵！你可以到我们的流量统计单去观察一下就知道啦！画面也是很不错的啰！

linux.vbird.org 主机流量统计表：
http://linux.vbird.org/flow/webalizer/index.html

如此一来，你就可以很轻松的观察你的主机的任何信息啰！包括来自于其它地方的 IP ，网页浏览数等等的！很棒吧！

---

awstats 网页分析利器：

- 官方网站：http://awstats.sourceforge.net/
- 设定难度：较难，需要有点技巧！
- 软件特色：中文化的很完整，而且该有的都有了，相当炫的一个分析利器！
- 授权模式：GPL

这套软件相当的不错！很棒，他提供了 CGI 程序执行与指令列模式执行，不过，我个人不太喜欢使用 CGI 的模式，所以我是使用指令列模式来进行这个程序的图形制作的！如果你是由官方网站下载新的版本来安装的话，那么底下的流程可能不是很适合你，目前我已经将一些需要修正的项目变更过，并且摆在我的网页上面，如果有需要的话，可以到『http://linux.vbird.org/download/index.php#awsats』来下载喔！

```
1. 下载软件：可以直接到官方网站，或者到我们网站上下载：
  http://linux.vbird.org/download/index.php#awsats
# 使用浏览器将网站上面提供的数据拿回去！当然，你也可以直接到官方网站上面去下载最新的版本！
# 假设你已经将数据捉回去，并且放置在 /root 这个目录当中了，文件名为：
awstats.tar.gz

2. 安装软件：
# 由于我已经将数据都丢在一起了，所以请到 /usr/local/apache2/htdocs 或者其它浏览器可以
# 进入浏览的目录下面，直接将上面捉到的档案解压缩即可！
[root@test root]# cd /usr/local/apache2/htdocs
[root@test htdocs]# tar -zxvf /root/awstats.tar.gz
# 会产生一个名为 awstats-5.3 的目录喔！

3. 设定文件编修：
# 这个设定档比较奇怪一点，他一定是 awstats.{your.hostname}.conf 的格式，所以，以我为例，
# 我的主机名称为 test.vbird.idv.tw 好了，那么就可以取名为 test ，所以，
[root@test htdocs]# cd /usr/local/apache2/htdocs/awstats-5.3
[root@test awstats-5.3]# cp awstats.HOSTNAME.conf awstats.test.conf
[root@test awstats-5.3]# vi awstats.test.conf
```

```
# 只要编辑前面几行就够了！
LogFile="/var/log/httpd/access_log" # Apache 的登录档，请依你的设定写入！
SiteDomain="test.vbird.idv.tw" # 你的主机名称，请修改吧！
HostAliases="localhost 127.0.0.1 192.168.1.2"       # 主机还有别名的话，请将他加入！
DirIcons="/awstats-5.3/icons" # The output's icons
DirCgi="/awstats-5.3"
DirData="/usr/local/apache2/htdocs/awstats-5.3/data"    # The output directory
DNSStaticCacheFile="/usr/local/apache2/htdocs/awstats-5.3/cache/dnscache.txt"
DNSLastUpdateCacheFile="/usr/local/apache2/htdocs/awstats-5.3/cache/dnscachelastupdate.txt"
# 再改一个小地方
[root@test awstats-5.3]# vi awstats.sh
cd /usr/local/apache2/htdocs/awstats-5.3
./awstats.pl -config=test -update -output > awstats.html
# 路径名称与 config 后面接的你刚刚 copy 的那个档案的档名(中间部分的名称)

4. 修改一下 httpd.conf 的设定内容：
[root@test awstats-5.3]# vi /usr/local/apache2/conf/httpd.conf
# 在最后一行给他加入底下的咚咚：
<Directory "/usr/local/apache2/htdocs/awstats-5.3">
    AddHandler cgi-script .cgi .pl
    AllowOverride AuthConfig
    Options +ExecCGI
</Directory>
[root @test awstats-5.3]# /usr/local/apache2/bin/apachectl stop
[root @test awstats-5.3]# /usr/local/apache2/bin/apachectl start

5. 测试 Run 的结果
[root@test awstats-5.3]# cd /usr/local/apache2/htdocs/awstats-5.3
[root@test awstats-5.3]# ./awstats.sh
[root@test awstats-5.3]# ls -l data
-rw-rw-rw-    1 root     root         4776 Jan 10 14:46 awstats012003.test.txt
# 如果有看到类似上面的档案出现在 data 当中，哈哈！那就是成功啦！

6. 设定每日执行！
[root @test root]# vi /etc/crontab
# 加入底下这一行：
25 03 * * * root /usr/local/apache2/htdocs/awstats-5.3/awstats.sh
```

这样也就修改妥当啰！详细的图形示意图可以参考：
http://awstats.sourceforge.net/cgi-bin/awstats.pl 相当的不赖吧！赶快试看看！

虚拟主机架设：

- 什么是虚拟主机 (Virtual Host)?
  咦! 虚拟主机是什么东西呐! 怎么说的好像很神奇的样子呢? 他有什么功能? 为什么大家都想要玩一下虚拟主机呐? 呵呵! 所谓的虚拟主机，基本上，就是『让你的一部主机上面，有好多个"主网页"存在，也就是说，硬件实际上只有一部主机，但是由网站网址上来看，则似乎有多部主机存在的样子!』，举个例子来说好了，你可以由我的网页上面知道一件事情，那就是我的 WWW 主机其实只有一部，那就是 pc510.ev.ncku.edu.tw 。不过，我这部主机其实有很多个 Domain name 存在，例如 linux.vbird.idv.tw 与 phorum.vbird.idv.tw 这两个网址。不过，上面这两个网址其实也同时指到 pc510.ev.ncku.edu.tw 这部机器上面，亦即是 pc510.ev.ncku.edu.tw, linux.vbird.idv.tw, phorum.vbird.idv.tw 这三个不同的主机名称，其实都是指向同一部计算机主机! 你可以在你的 Linux 机器上面以 nslookup 的指令搜寻一下这三个主机名称，你就晓得为什么了! 但是，当你在浏览器上面输入:

  - http://pc510.ev.ncku.edu.tw
  - http://linux.vbird.org
  - http://phorum.vbird.org

  怪了! 怎么会显示不同的网页内容呢? 好像是存在三部主机对吧! 呵呵! 这就是所谓的虚拟主机啦! 让你的一部机器上面，搞的好像好多部同的主机一样的一个简易的功能啦! ^_^

- 架设的大前提:
  那么要架设虚拟主机需要什么咚咚呢? 呵呵! 以刚刚我的网站的结果为例，我要架设三个主网页，也就是必须要有多个 domain name 啰! 对啦! 这就是虚拟主机的大前提啦! 『你必须要有多个主机名称，亦即是需要多个 domain name, FQDN 』，例如上面我就具有三个 domain name 啰! 在需要多个 domain name 的情况下，你可以做的方式就是:

  - 申请多个 host name 在 ISP 的管理上面;
  - 自行设定经过合法授权的 DNS 主机来自行设定自己的 domain name !

  没错，就是这几个方法，因此，请确定你的主机名称已经搞定了! 不然的话，怎么架设虚拟主机呢? 您说是吧! ^_^

- 实际架设:
  好了! 又要来搞设定啦! 又是 httpd.conf 这个档案啦! 反正几乎只要跟 Apache 有关的，就是动这个档案就对啦! 这里先来说一下我的大前提设定啰!

  - 已经设定好了三个 domain name ，分别是 mdk90.vbird.org, www.mdk90.vbird.org, phorum.mdk90.vbird.org ，此外，这三个网域的主网页个别放置在 /home/mdk90, /home/www.mdk90, /home/phorum.mdk90 ，亦即是:
    - mdk90.vbird.org --> /home/mdk90
    - www.mdk90.vbird.org --> /home/www.mdk90
    - phorum.mdk90.vbird.org --> /home/phorum.mdk90

至于这个设定嘛！真的是很简单耶！只要几行就搞定了，设定完成之后还会让你偷笑ㄋㄟ～呵呵！看看实例吧！

```
[root@test root]# cd /usr/local/apache2/conf
[root@test root]# vi httpd.conf
# 在这个档案的最下方加入底下这些字眼！

NameVirtualHost *          # 设定你的虚拟主机判定的依据！这里是* 亦即是
                           # 所有连上这部机器的名称都会被使用来当作虚拟主机的设定之用！

<VirtualHost *>
    ServerName     mdk90.vbird.net
    DocumentRoot   /home/mdk90
</VirtualHost>

<VirtualHost *>
    ServerName     www.mdk90.vbird.net
    DocumentRoot   /home/www.mdk90
    CustomLog      /var/log/httpd/www.access_log combined # 特别将登录档额外分离出来
</VirtualHost>

<VirtualHost *>
    ServerName     phorum.mdk90.vbird.net
    DocumentRoot   /home/phorum.mdk90
</VirtualHost>
```

要注意的是：

1. 在虚拟主机的设定上还有很多的可用的功能，不过，最低的限度是需要有 ServerName 及 DocumentRoot 这两个即可！
2. 虽然原来我就有 mdk90.vbird.net 这个网域，但是因为设定了虚拟主机之后，自己的原来名称可能会不见去，所以，这里必须将自己的名称也写入才行！
3. 上面有发现一个 CustomLog 的设定喔！该设定会让以 www.mkd90.vbird.net 这个网域登入的登录档不再写入原来的 /var/log/httpd/access_log 档案，而是自行写入 /var/log/httpd/www.access_log 这个档案！

马上测试看看！呵呵！会发现，咦！我真的有三个主网页了哩！很不错吧！这个作法可以让你的 WWW 网页更有灵活度喔！举个例子来说，前一阵子因为酷学园讨论区(http://phorum.study-area.org) 常常挂点，所以鸟哥就自告奋勇的跟站长说，只要将 phorum.study-area.org 这个 domain name 指向我的主机 IP ，那么也可以直接进入我的讨论区，如此一来，只要修改一下 DNS 即可转换到我的讨论区啦！并且不需要再进行任何额外的设定！对于网页维护的灵活度是真的很有帮助的喔！

- 需要注意的事项：

虚拟主机并没有什么值得特别注意的地方，只要设定正确，大致上就不会有太大的问题！不过，你可能需要特别注意刚刚我们建立起来的新的登录档喔！为什么呢？我们上面不是提过说，登录档在大型的网站上面成长的幅度是很可观的吗？所以需要进行 logrotate ，但是你刚刚建立的档案并不在原本的 logrotate 档案之内呀！呵呵！这个时候请自行加入 logrotate 个手续喔！否则.....嘿嘿嘿嘿！硬盘空间被用光可不要怪鸟哥喔！ ^_^

客户端的文字接口 Web 功能：
什么！？客户端竟然也有文字接口的浏览器？！哈哈！当然是有啦！不然这里干嘛要介绍？！那就是鼎鼎大名的 lynx 以及 wget 啰！请注意的是，这两个套件并不一定会在安装的时候就已经安装在你的系统中，所以请先使用 RPM 查询一下他是否存在于你的系统当中，然后才能执行喔！他的用途是：

- lynx：文字接口的浏览器，相当的轻巧与快速！
- wget ：文字接口下使用来撷取档案的指令！

这两个指令之前已经介绍过了！请自行前往观察一下啰！加油啦！

增强 PHP 程序代码执行速度的模块：

MM Cache 增强速度模块：

我们上面的介绍都是在于安装与架设 LAMP 服务器(Linux + Apache + MySQL + PHP)，不过，如果您曾经浏览过 PHP 的网页时，或许会发现：『咦！怎么 PHP 的速度慢慢的.....』，这是怎么一回事啊!？PHP 不是号称速度上面的反应是很快速的吗？怎么会慢慢的呢？虽然 PHP 的程序已经很快了，但是因为计算机仅认识 0 与 1 的 binary file 来执行，而由于 PHP 程序不需要编译即可透过 PHP 核心与其相关函式库来执行，不过，如此一来还是多了一道手续，导致执行效能还是不比传统的经过编译的程序语言来的快(例如C程序语言之类的！)。那怎么办？有办法加快 PHP 程序的速度吗？嗯！让我们先换个角度想，如果我们可以将 PHP 程序预先转换成为可直接执行的 binary file ，那么不就可以直接读取，进而加快速度吗？没错！是这样～这东西称为预编器～其中，MM Cache 就是一个很不错的 PHP 预编器。MM Cache 可以将您的 PHP 程序与 PHP 核心及相关函式库预先编译后暂存下来，以提供未来使用时，可以直接执行，加上他可以优化您的 PHP 程序，因此，可以让您的 PHP 网页速度增快不少喔！ MM Cache 啰！他的官方网站在：http://turck-mmcache.sourceforge.net/index_old.html。目前 MM Cache 已经出到了 2.3.9 (2003/04/10 释出)，不过，他仅在 PHP 4.1 以上版本以及 Apache 1.3 与 2.xx 版本测试过，如果你的 Apache 与 PHP 不是这些版本，那就抱歉啦！不一定可以使用的！不过，呵呵！我们上面提供的 Tarball 的安装方法本来就是这样的版本，所以您可以轻易的安装好 MM Cache 喔！废话不多说，赶紧来安装吧！

- 下载：
  你可以在官方网站下载最新的版本 (http://turck-mmcache.sourceforge.net/index_old.html)，也可以在我们网站下载鸟哥试过的版本：
  http://linux.vbird.org/download/index.php#www_mmcache

- 安装：
  安装 MM Cache 真的是很简单喔！赶紧来安装吧！

```
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /完整路径/turck-mmcache-2.3.9.tar.gz
# 会产生一个名为 turck-mmcache-2.3.9 的目录

[root@test src]# cd turck-mmcache-2.3.9
[root@test turck-mmcache-2.3.9]# phpize
# 这个指令是 PHP 套件所提供的！可以建置好你的 mmcache 原始码

[root@test turck-mmcache-2.3.9]# ./configure --enable-mmcache=shared
[root@test turck-mmcache-2.3.9]# make && make install
# 这个动作会编译一个名为 mmcache.so 的动态函式库模块，
# 并且会主动的将他安装在 /usr/lib/php4 这个目录当中！
# 这样就安装完毕了！很简单吧！
```

-
- 设定：
  在 MM Cache 的设定方面需要更动两个地方，第一个是动态函式库加载的设定，第二个则是 PHP 的设定！

```
1. 设定主动加载动态函式库模块：
[root@test root]# vi /etc/ld.so.conf
# 在这个档案内加入底下这一行：
/usr/lib/php4

[root@test root]# ldconfig
# 上面这两个步骤比较有趣一点，在 ldconfig 这个指令的功能是：
# 『加载动态函式库到内存当中做为快取』之用，至于加载的动态函式库则是
# 根据 /etc/ld.so.conf 这个档案的设定为准！这的动作只要第一次设定时进行
# 即可，未来在开机完成之后，系统会主动的加载动态函式库的！
# 另外请注意， ld.so.conf 里面只要写『目录』即可！

2. 修改 php.ini
# 请注意，由于每个人的 php.ini 都不相同，例如使用 RPM 安装者，应该是
# /etc/php.ini ，但是我上面的安装设定却是 /usr/local/php4/php.ini
# 请依照您的主机来设定喔！
[root@test root]# vi /完整路径/php.ini
# 在这个档案的最后一行加入底下这几行：
;;;;;;;;;;;;
; MM Cache ;
;;;;;;;;;;;;
extension="mmcache.so"
```

```
mmcache.shm_size="16"
mmcache.cache_dir="/tmp/mmcache"
mmcache.enable="1"
mmcache.optimizer="1"
mmcache.check_mtime="1"
mmcache.debug="0"
mmcache.filter=""
; end of mmcache

3. 建立快取目录：
[root@test root]# mkdir /tmp/mmcache
[root@test root]# chmod 0777 /tmp/mmcache

4.重新启动 Apache
[root@test root]# /etc/rc.d/init.d/httpd restart
# 或
[root@test root]# /usr/local/apache2/bin/apachectl restart
```

- 这样一个简单的小步骤，嘿嘿！您的 PHP 程序代码的反应性～啊～增快很多很多喔！^_^

Apache 的效能测试

事实上，安装 Apache 的时候，Apache 就已经提供了一个效能测试（ benchmark ）的软件了！那就是 ab 这个程序！怎么用呢？就直接用啊！

```
[root@test root]# /usr/sbin/ab [-dSk] [-c number] [-n number] 网页.php
参数说明：
-d ：不要显示 saved table 的百分比资料；通常不要那个数据，所以会加 -d
-k ：还记得上面的 KeepAlive 吧！加入 -k 才会以这样的功能测试；
-S ：不显示长讯息，仅显示类似 min/avg/max 的简短易懂讯息！
-c ：同时有多少个『同时联机』的设定(可想成同时联机的 IP )
-n ：同一个联机建立几个要求通道！(可想成同一个 IP 要求的几条联机)
更多的讯息请自行 man ab 喔！
范例：
[root@test root]# /usr/sbin/ab -dSk -c100 -n100 \
> http://linux.vbird.org/home.php
This is ApacheBench, Version 1.3d <$Revision: 1.67 $> apache-1.3
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright (c) 1998-2002 The Apache Software Foundation, http://www.apache.org/

Benchmarking linux.vbird.org (be patient).....done
Server Software:        Apache/1.3.27
```

```
Server Hostname:        linux.vbird.org
Server Port:            80

Document Path:          /home.php
Document Length:        51736 bytes

Concurrency Level:      100
Time taken for tests:   1.648 seconds
Complete requests:      100
Failed requests:        0
Broken pipe errors:     0
Keep-Alive requests:    0
Total transferred:      5795454 bytes
HTML transferred:       5775070 bytes
Requests per second:    60.68 [#/sec] (mean)
Time per request:   1648.00 [ms] (mean)
Time per request:   16.48 [ms] (mean, across all concurrent requests)
Transfer rate:      3516.66 [Kbytes/sec] received

Connnection Times (ms)
              min   avg   max
Connect:       52   309   369
Processing:   467   687  1079
Total:        467   996  1405
```

这样就能够了解您的 Apache 效能了！因为我是在本机上面测试的，所以速度上面当然是很快啰！建议可以到远程同样使用 ab 来测试一下你的 Apache 效能，尤其是加上了 mm cache 之后，看看能不能增快速度呢？（ 注：这个 ab 程序对于读取 MySQL 之类的网页似乎没有办法成功的完成测试的样子，所以请不要使用 phpBB2 的网页来测试喔！尽量直接以 PHP 的网页来测试！ ）嘿嘿！

砍站软件与 Nimda 病毒的抵挡 scripts：

几个比较知名的网站管理员大概都有这样的困扰，那就是网站常被砍站软件所强力下载，结果造成主机的 CPU loading 过重，最后竟然会导致死掉～唉！真是的～人怕出名猪怕肥呐！先来解释一下什么是砍站吧！所谓的『砍站』，就是以类似多点联机下载的持续性讯息传递软件进行网站数据的下载，而且，一启用该软件，该软件就将『整个网站』的内容都给他 download 下来，很厉害吧！没错！是很厉害，但是却也害死人了～怎么说呢？因为这种软件常常会为了加快 download 的速度，所以采用多点联机的方式，也就是会持续不断的向 Server 发出要求封包，而由于这些封包并不见得能够成功的让 Server 把数据传导给 Client 端，常常会无法投递就是啦！这样的结果就是.....造成 Server 要一直不断的响应，又无法正确的响应出去，此外，要求太过频繁，结果主机应接不暇，最后....就当机了...真的是林老师ㄌㄟ～我们这个小站的主机古早以前，就是因为这样的原因，导致服务常常断断续续的，并且，由于 CPU loading 太高，结果让正常联机进来看数据的网友没有足够的资源，因此网页开启的速度就变的很慢～唉～这些砍站的人，也太不道德啦！

由于这种砍站软件真的很麻烦，一不注意马上就又会被砍站而当机，三天两头就要重新开机一次，完全让 Linux 的稳定性无法发挥！真是气死了～后来，我就自行写了一个 scripts 来挡这样的 IP ！我的作法是这样的：

1. 由于砍站软件要多点连续下载，因此，同一个 IP 在同一个时间内，会有相当多的联机发生；
2. 由于他是重复不断的要求联机，因此刚刚建立的联机在达成下载的目的后，会立刻死掉，而又多生出其它的联机出来，因此，这个时候他的联机情况就变的相当的不正常了！
3. 由于某些较旧的砍站软件并不会『欺骗』主机，所以，会在主机的登录文件里面记录住 Teleport 的标记！
4. 既然如此的话，那么我就让我的主机每分钟去检查两个东西(1)先检查 log file ，如果有发现到相关的 Teleport 字词，就将该 IP 抵挡掉；(2)使用 netstat 来检查同一个 IP 的同时联机，如果该联机超过一个值(例如同时有 12 个联机)的话，那么就将该 IP 抵挡掉！
5. 此外，由于上面的方案可能会将 Proxy 的 Client 端也同时抵挡掉，真是可怜啊！这个时候，这支程序就会主动的将(1)的情况的主机抵挡 3 天，至于(2)的情况则抵挡 2 小时！过了该抵挡的时限后，该 IP 即可又连上我们的主机了！

大致上就是这样吧！这样的一程序需要与 iptables 相互配合，所以，请先查阅一下简易防火墙设定那一篇文章，然后再来下载这支程序吧！这支程序您可以在底下的网址下载喔！
http://linux.vbird.org/download/index.php#http-netstat.sh
详细的安装步骤我已经以中文写在该档案里面了，所以请先查看一下该档案的前面说明部分吧！此外，Study Area 的 netman 大哥也已经开发了一套很棒的防砍站的程序了！在防堵砍站的原理上面是完全相同的，不过写法可能不是很雷同就是了！如果有需要的话，也可以前往 Study-Area 搜寻一下啰！
http://phorum.study-area.org/viewtopic.php?t=13643

安装 phpBB2 讨论板：
上面这样一路走来，哈哈！终于我们的 LAMP 服务器就已经大致上搞定啦！那么接下来你可以利用这个 WWW 主机帮你做什么事呢？嘎！能作的事情可多啰！目前很多支持 PHP 的架站软件已经被很完整的开发了，例如 PHPNuke 以及鸟哥很喜欢的 phpBB 呢！这些架站软件都是建构在 LAMP 上面的，而既然我们的 LAMP 已经搞定了，那么其它的架站软件的安装就真的是相当的快速呢！底下介绍 phpBB 的安装！你可以到底下的连结去看看喔：

- phpBB 官方网站：http://www.phpbb.com/
- phpBB 正体中文网站『竹猫星球』：http://phpbb-tw.net/phpbb/
- 简易 phpBB2 的安装与设定方法：http://linux.vbird.org/apache_packages/

上面最后一个是鸟哥前一阵子写的，目前已经有出较新版本的 phpBB2 啰！所以，请记得到官方网站下载最新的 phpBB 来安装喔！毕竟比较新的不但功能比较多，而且臭虫(Bug)也清理的差不多了！另外，由竹猫星球的竹猫三兄弟也有出一本『phpBB 论坛架设宝典』，里面也有提到相当多的有用的架站心得与技巧的说明！有兴趣的可以先到竹猫星球看看其风格与内容，然后再考虑要不要架站吧！^_^目前我对 phpBB 倒是蛮喜欢的！

问题讨论：

- 怪了！怎么我按照上面文章的设定，设定了底下的咚咚，但是却还是无法显示中文？为什么？
  AddDefaultCharset Big5

LanguagePriority tw en da nl et fr de el it ja ko no pl pt pt-br ltz ca es sv
答：

一般而言，按照上面的设定来编写 httpd.conf 这个档案的内容之后，就可以正确无误的显示中文字码才对～但是很多朋友还是来信说到,怪了!怎么会还是无法显示中文呢?研究了一下其中的原因，发现其实大家都有个错误的用法啦！就是因为，目前的浏览器上面都有所谓的快取 （ Cache ） 功能，这个功能主要会将浏览过的网页及其相关信息，例如图片、与最重要的网页的纪录等，先以档案的型态储存在计算机的硬盘中，若下次您在短时间内拜访该网站时，那么浏览器将会直接由硬盘 （Cache） 将数据拿出来使用，所以就不会再次到远方读取数据！这可以节省相当多网络流量频宽，以及缩短等待的时间喔！不过，也由于如此，所以，你上次看到显示乱码的那个网页，也就会被储存在硬盘当中，因此，在你修改完成 httpd.conf 并且重新启动 Apache 之后，再次浏览该网页，仍然可能无法显示出正确的中文的啦！就是因为你曾经浏览过啦！那么要如何解决呢？

1. 最简单的方法就是按下『Reload, 重新读入』，不过，因为有的浏览器是藉由『比对网页数据与自己储存的 cache 数据是否相同』来做为 cache 更新的依据，而由于你仅修改了 httpd.conf 这个档案，对于网页并没有关系，因此，可能还是会显示出不正确的中文字喔！
2. 再来的方法就是『改文件名称！』，既然名称已经跟硬盘的纪录不一样了，因此，该笔数据就会被重新读取，自然就会更新啰！
3. 不过，最有效的方法，还是直接将你的浏览器内的高速缓存 （ Cache ） 清除掉就可以啦！以 IE 为例：点选『工具』选择『Internet 选项』，按下『删除档案』就可以啦！不过，如此一来，你曾经浏览过的网页的快取数据同时也会被清除掉就是了！

- 伤脑筋呐！怎么完全按照上面文章的方法，结果 Apache 与 MySQL 是启动了，但是 PHP 就是无法启动？害得我的 PHP 老是直接显示出原始的档案内容，而不是执行呢？
答：

这个问题并不会存在于旧版的 PHP 当中,只有新版的 PHP 才有此一问题!这与 php.ini 的设定有关啦其实，与 Apache 的设定可就没有关系啦！解决的方法可以有两种方式：

1. 直接在你的 php 档案中，在 <? 的地方，都改写成 <?php 就可以啦！例如

```
<?
phpinfo( );
?>
```

2. 改成

```
<?php
phpinfo( );
?>
```

3. 就可以啦！
4. 如果不想要这么麻烦的话，可以直接修改 PHP 的设定档，亦即是 php.ini 这个档案，在我们上面文章的例子中，使用的是 /usr/local/php4/php.ini 这个目录下的档案，但如果是一般 distribution 提供的 PHP 的话，例如 Red Hat 8.0 ，那么该档案将会放置在 /etc/php.ini ，总之，请以搜寻的指令找出该档案就对了！然后，以 vi 编辑该档案，找到 short_open_tag 那一段，将该设定改成：

   ```
   short_open_tag = On
   ```

   这样就可以不需要在 <? 右边加上 php 而可以执行 PHP 啰！

- 如何解决出现 MySQL 的这个问题：
  ERROR 2002: Can't connect to local MySQL server through socket '/tmp/mysql.sock' (111)
  答：

  这个问题多半出现在找不到 mysql.sock 这个档案，所以：

  1. 请先确认 MySQL 已经正确的启动当中(netstat -tl 找寻看看)；
  2. 再确定 /tmp/mysql.sock 是否真的存在，若不存在，请使用 find / -name mysql.sock 找到这个档案，假设这个档案完整路径在 /var/lib/mysql/mysql.sock 好了；
  3. 再使用 mysqladmin -S /var/lib/mysql/mysql.sock -u root {其它参数} ...
  4. 测试结果应该都可以正确的启动了才对！

- 当我在启动 /etc/rc.d/init.d/httpd restart 时，总是会出现如下的问题：

```
[root@test root]# /etc/rc.d/init.d/httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd: perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = "en",
        LC_ALL = "en",
        LANG = "en"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
```

- 虽然对于我的 Apache 不会产生什么不良的影响，但是总觉得不太舒服，要如何克服呢？
  答：
  - 因为您的 apache 有使用 perl 的模块，偏偏 perl 模块会读取 locale 这个咚咚的变量，这个 locale 应该就是跟语言有关的一些环境变量咯！而我们语言的环境变量与 i18n

有关，也就是在 /etc/sysconfig/i18n 里面的设定咯！如果是在 Red Hat 9 的系统中，与 i18n 有关的设定值在：『/usr/share/i18n/locale』当中，这里面有一些我们惯用的语言存在！一般来说，我们的语言通常预设为中文(zh_TW)或者是英文(en)，但是在 /usr/share/i18n/locale 当中并没有 en 存在(我的 red hat 9 没有...)，所以这个时候要以 en_US (因为是美语啊！)来设定即可！如何设定呢？

-

```
[root@test root]# vi /etc/sysconfig/i18n
LANG="en_US"
LANGUAGE="en_US"
LC_ALL="en_US"; export LC_ALL
SUPPORTED="zh_TW.Big5:zh_TW:zh:en_US.UTF-8:en_US:en"
SYSFONT="latarcyrheb-sun16"
```

- 这样应该就不会出现问题了吧？！

课后练习

- 请问 LAMP 这个服务器代表什么意思？
- Apache 的设定档档名一般为何？
- 在 Apache 的设定档当中，哪一个参数是用来设定『主网页』的？
- 哪一个指令用来重新启动与关闭 Apache ？(请以 Tarball 安装的方法来说明)
- 当我使用 ps -aux 的时候，发现好多的 httpd... 的程序，这是正常的吗？！最多可以有几个程序是在那个档案的那个参数所设定的？！
- 又，呈上题，这些程序 (process) 的 owner 与 group 是谁？该察看那个设定档的那个参数？
- 如果今天我以 http://your.ip 结果却发现浏览器出现类似 FTP 的画面(会列出该目录下的所有档案)，这是什么原因造成的？该如何避免？
- 在 Apache 里面， .htaccess 这个档案的功能为何？

简易 SAMBA 服务器设定

最近更新日期：2003/09/30

在一般的局域网络中 (LAN) 如果都是 Windows 计算机，那么使用『网络上的芳邻』这个功能，就可以让不同的 Windows 计算机分享彼此的档案啰！那么万一这个 LAN 里面有个 Linux 主机时，我怎么让 Linux 也加入这个 Windows 计算机当中的『网络上的芳邻』呢？！也就是说，让 Windows 计算机可以透过『网络上的芳邻』来存取 Linux 主机上面的档案！呵呵！那就是 SAMBA 这个服务器的主要目的了！SAMBA 是很有用的一个服务器，他可以让您的 Linux 刹那间成为一个档案服务器 (File Server)，并提供整个 LAN 里面的 Windows 很简单的就可以对 Linux 主机进行档案的存取动作。不仅如此， SAMBA 也可以让 Linux 上面的打印机成为打印机服务器 (Printer Server) ，鸟哥个人觉得， SAMBA 对于整个 LAN 的贡献真的是很大！那么就赶紧来玩一玩吧！ ^_^

原理：

在这个章节中，我们要教大家跳的是热情有劲的巴西 SAMBA 舞蹈.....喔不～搞错了～是要向大家介绍 SAMBA 这个好用的服务器啦！咦！怪了！怎么服务器的名称会使用 SAMBA 呢？还真是怪怪的呢！那么这个 SAMBA 服务器的功能是什么呢？另外，他最早是经由什么样的想法而开发出来的呢？呵呵！底下就让我们慢慢的谈一谈吧！

SAMBA 的发展历史与名称的由来

在早期的网络世界当中，档案数据在不同主机之间的传输大多是使用 FTP 这个好用的服务器软件来进行传送。不过，使用 FTP 传输档案却有个小小的问题，那就是您无法直接修改主机上面的档案数据！也就是说，您想要更改 Linux 主机上面的某个档案时，必需要由 Server 端将该档案下载到您工作的 Client 端后才能修改，也因此该档案在 Server 与 Client 端都会存在。这个时候，万一如果有一天您修改了某个档案，却忘记将数据上传回主机，那么等过了一阵子之后，呵呵，您如何知道那个档案才是最新的？！

既然有这样的问题，那么好吧，我可不可以在 Client 端的机器上面直接取用 Server 上面的档案，如果可以在 Client 端直接进行 Server 端档案的存取，那么我在 Client 端就不需要存在该档案数据啰，也就是说，我只要有 Server 上面的档案资料存在就可以啦！有没有这样的档案系统啊（ File System ），呵呵！很高兴的是，前面我们已经提过的 Network File System, NFS 就是这样的档案系统之一啦！我只要在 Client 端将 Server 所提供分享的目录挂载进来，那么在我 Client 的机器上面就可以直接取用 Server 上的档案资料啰，而且，该数据就像是我 Client 端上面的 partition 一般，真是好用！而除了可以让 Unix Like 的机器互相分享档案的 NFS 服务器之外，在微软（ Microsoft ）上面也有类似的档案系统，那就是 Common Internet File System, CIFS 这个咚咚啦！CIFS 最简单的想法就是目前常见的『网络上的芳邻』咯！ Windows 系统的计算机可以透过桌面上『网络上的芳邻』来分享别人所提供的档案数据哩！真是方便。不过，NFS 仅能让 Unix 机器沟通， CIFS 只能让 Windows 机器沟通。伤脑筋，那么有没有让 Windows 与 Unix-Like 这两个不同的平台相互分享档案数据的档案系统呢？

在 1991 年一个名叫 Andrew Tridgwell 的大学生就有这样的困扰，他手上有三部机器，分别是跑 DOS 的个人计算机、DEC公司的 Digital Unix 系统以及 Sun 的 Unix 系统。在当时， DEC 公司有发展出一套称为 PATHWORKS 的软件，这套软件可以用来分享 DEC 的 Unix 与个人计算机的 DOS 这两个操作系统的档案数据，可惜让 Tridgwell 觉得较困扰的是，Sun 的 Unix 无法藉由这个软件来达到数据分享的目的。这个时候 Tridgwell 就想说：『咦！既然这两部系统可以相互沟通，没道理 Sun 就必需这么苦命吧？可不可以将这两部系统的运作原理找出来，然后让 Sun 这部机器也能够分享档案数据呢？』，为了解决这样的的问题，他老兄就自行写了个 program 去侦测当 DOS 与 DEC 的 Unix 系统在进行数据分享传送时所使用到的通讯协议信息，然后将这些重要的信息撷取下来，并且基于上述所找到的通讯协议而开发出Server Message Block (SMB) 这个档案系统，而就是这套 SMB 软件就能够让 Unix 与 DOS 互相的分享数据啰！（ 注：再次的给他强调一次，在 Unix Like 上面可以分享档案资料的 file system 是 NFS，那么在 Windows 上面使用的『网络上的芳邻』所使用的档案系统则称为 Common Internet File System, CIFS ）

既然写成了软件，想一想，嗯！总是需要注册一下商标吧！因此 Tridgwell 就去申请了 SMBServer

(Server Message Block 的简写) 这个名字来做为他撰写的这个软件的商标，可惜的是，因为 SMB 是没有意义的文字，因此没有办法达成注册。既然如此的话，那么能不能在字典里面找到相关的字词可以做为商标来注册呢？翻了老半天，呵呵！这个 SAMBA 刚好含有 SMB ，又是热情有劲的拉丁舞蹈的名称，不然就用这个名字来做为商标好了！哈哈！这成为我们今天所使用的 SAMBA 的名称由来啦！ ^_^

SAMBA 的应用功能

由上面说明的 SAMBA 发展缘由，您就应该不难知道咯， SAMBA 最初发展的主要目就是要用来沟通 Windows 与 Unix Like 这两个不同的作业平台，这么做有什么好处呢？刚刚我们上面不就已经说过了，最大的好处就是您不必让同样的一份数据放置在不同的地方，搞到后来都不晓得哪一份资料是最新的！而且也可以透过这样的一个档案系统上 Linux 与 Windows 的档案传输变得更为简单！也就是说，您以后可以透过『网络上的芳邻』来进行 Linux 与 Windows 档案的传输啦！那么 SAMBA 可以进行哪些动作呢？

- 分享档案与打印机服务；
- 可以提供使用者登入 SAMBA 主机时的身份认证，以提供不同身份者的个别数据；
- 可以进行 Windows 网络上的主机名称解析 (NetBIOS name)
- 可以进行装置的分享 (例如 Zip, CDROM...)

底下我们来谈几个 SAMBA 服务器的应用实例吧！

应用实例一：以鸟哥为例，由于我都是使用 Windows 系统来编辑我的网页画面，然后再传到我的 Linux 机器上。一开始，鸟哥也是以 FTP 来传送我的网页的，后来发现，这样在我的 Windows 上面需要有一份网页数据，然后修改完成之后又要传到 Linux 上面，如此便有两个相同的档案，最麻烦的是，有时候下载下来的档案已经经过好多修改了，却在下次的 FTP 作业，不小心又下载一次旧数据，结果将已经修改过的数据覆盖过去～天呐！又要重写一遍.....真是讨厌！后来，鸟哥就安装了 SAMBA 服务器，将 Linux 上我的网页目录打开成可以资源共享，如此一来，鸟哥就可以直接透过 Windows 的『网络上的芳邻』来修改我的网页数据啰！而且，这就有点像是『在线编修』呢，一修改完成，在 Internet 上面可以立刻检验，方便的很呐！

应用实例二：在我们实验室中，由于计算机数量不多，研究生常常会使用到不同的计算机 (因为大家都得抢没有人用的计算机啊！)，此外，也常常有研究生拿自己的 NoteBook 来工作，因此，有些团队的数据就分散在各个计算机当中，使用上相当的不方便。这个时候，我就使用 SAMBA 将硬盘空间分享出来，由于使用者要登入 SAMBA 这个服务器主机时需要输入使用者数据 (账号与密码)，而不同的登入者会取得不一样的目录资源，所以，可以避免自己的数据在公用计算机上面被窥视，此外，在不同的公用计算机上面都可以登入 SAMBA 主机，数据的使用上面真是相当的棒啊！

应用实例三：SAMBA 除了分享档案系统外，也可以分享打印机喔，我们研究室好几部计算机就是直接以 Linux 分享的打印机来印制报告的。您会说『啊 Windows 也可以办的到啊！没有什么了不起的！』是啊。但是因为 Linux 做为服务器主机时，鸟哥认为 Linux 毕竟还是比较稳定一点，

可以 24 小时且全年无休的努力工作呐。此外，因为目前透过『网络上的芳邻』来攻击局域网络的 Windows 操作系统的计算机病毒实在是太多了，防不胜防， Linux 对于这样的攻击并没有很大的影响 (因为常见的攻击手法均针对 Windows 而来～)，所以也比较安全一些说～

SAMBA 的应用挺广泛的，尤其对于局域网络内的计算机来说，更是一项不可多得的好用的服务器，更多的应用您可以自行发掘呐!

NetBIOS 通讯协议

事实上，就像 NFS 是架构在 RPC Server 上面一样， SAMBA 这个档案系统是架构在 NetBIOS (Network Basic Input/Output System, NetBIOS)这个通讯协议上面所开发出来的。既然如此，我们当然就要了解一下 NetBIOS 啰!最早 IBM 发展出 NetBIOS 的目的仅是要让局域网络内少数计算机进行网络连结的一个通讯协议而已，所以考虑的角度并不是针对大型网络，因此，这个 NetBIOS 是无法跨路由的 (Router/Gateway)。这个 NetBIOS 在局域网络内 (Local Area Network, LAN) 实在是很好用,所以微软的网络架构就使用了这个咚咚来进行沟通的呐!而 SAMBA 最早发展的时候，其实是想要让 Linux 系统可以加入 Windows 的系统当中来分享使用彼此的档案数据的，所以当然 SAMBA 就架构在 NetBIOS 发展出来啰。

不过，如果单纯的使用 NetBIOS 而已，偏偏 NetBIOS 是无法跨路由的，那么该服务器的使用范围不就受限相当的多了？好在，我们还有所谓的 NetBIOS over TCP/IP 的技术呢！这是什么样的技术啊？！举个例子来说好了，我们知道 TCP/IP 是目前网络连接的基本协议，现在，我们将 NetBIOS 想成是一封明信片，这个明信片只能让您自己欣赏而已，如果今天我们要将这个明信片送到远方的朋友那边时，呵呵！就需要透过邮件系统 (例如邮局啦、国际快递啦等等的) 来传送了！这个 TCP/IP 就可以视为邮件传递系统啦！透过这个 NetBIOS over TCP/IP 的技术，我们就可以跨路由的使用 SAMBA 服务器所提供的功能咯！当然啦，目前 SAMBA 还是比较广泛的使用在 LAN 里面说。

注：或许您会发现在 Windows 网络设定里面常常看到 NetBEUI 这个咚咚，那是什么呢？事实上，那个是 NetBIOS Extened User Interface 的简写，也是 IBM 在 NetBIOS 发展出来之后的改良版本。虽然这两者的技术不太相同，不过，我们只要知道一些简单的概念就可以了！所以，在这里我们不针对 NetBEUI 来介绍。

SAMBA 使用的 daemons

知道了 SAMBA 的主要目的是让 Linux 主机加入 Windows 的网络系统当中来分享使用彼此的数据，而 Windows 使用的是 NetBIOS 这个通讯协议，所以说，SAMBA 主要是使用 NetBIOS over TCP/IP 的技术。好了，我们再来谈一谈，那么 SAMBA 在 Linux 操作系统上面工作时，需要启用什么服务呢 (daemons)？让我们先以 Windows 的『网络上的芳邻』来做简单的说明：

- 当我们想要登入某部 Windows 主机使用他所提供的档案数据时，必需要加入该 Windows 主机的群组 (Workgroup)，并且我们的机器也必需要设定一个主机名称，注意喔，这

个主机名称跟 Hostname 是不一样的，因为这个主机名称是架构在 NetBIOS 协议上的，我们可以简单的称呼他为 NetBIOS Name 好了。在同一个群组当中，NetBIOS Name 必需要是独一无二的喔！

- 好了，等到我们登入该主机之后，能不能使用该主机所提供的档案数据还要看 Windows 主机有没有提供我们使用的权限呐！所以，并不是登入该 Windows 主机之后，我们就可以无限制的取用该主机的档案资源了。也就是说，如果对方主机允许你登入，但是却没有开放任何资源让您取用，呵呵，登入主机也无法查看对方的硬盘里面的数据的啦！

了解了响，同样的 SAMBA 主机就使用两个 daemons 来管理这两个不同的服务：

- smbd ：这个 daemon 的主要功能就是用来管理 SAMBA 主机分享什么目录、档案与打印机等等的内容。
- nmbd ：这个 daemon 则是用来管理群组啦、NetBIOS name 啦等等的解析。

所以啰， SAMBA 每次启动至少都需要有这两个 daemons 喔！这可不要忘记啰！ ^_^而当我们启动了 SAMBA 之后，主机系统就会启动 137, 138, 139 三个 port ，且同时会有 UDP/TCP 的监听服务喔！这可不要忘记了！因为后面设定防火墙的时候，还会使用到这三个 port 的呢！

联机模式的介绍 (peer/peer, domain model)

SAMBA 主机的应用相当的广泛，而且可以依照不同的网域联机与使用者账号、密码的控管方式不同，来加以不同的类别应用，例如最常见的 Workgroup 及 Domain 两种方式的联机模式呢！底下我们就是要来谈一谈这两种最常见的局域网络的联机模式 peer/peer 及 domain model。

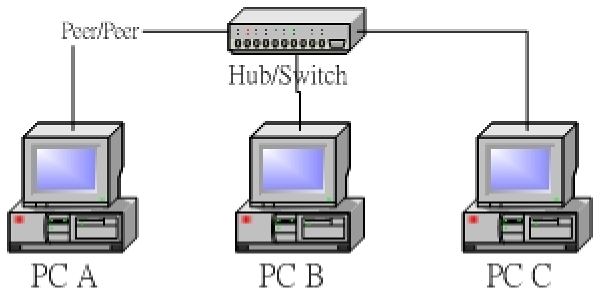
- peer/peer (Workgroup model) ：

  peer 有同等、同辈的意思存在，那么 peer/peer 由字面上的解释来看，当然就是同等地位的 PC 架构了！这是什么意思呢？简单的来说，在局域网络里面的所有 PC 均可以在自己的计算机上面管理自己的账号与密码，同时每一部计算机也都具有独力可以执行各项软件的能力，只是藉由网络将各个 PC 连结在一起而已的一个架构，所以，每一部机器都是可以独立运作的喔！而在这样的架构下，如果有两部计算机，计算机名称假设为 pc1 与 pc2 好了，那么当您要坐在 pc1 这部计算机前使用 pc1 的资源时，就必须要知道登入 pc1 的使用者名称与密码，才能够登入使用。而如果您想由 pc1 经过网络联机到 pc2 来使用 pc2 的档案资源时，就必须要知道 pc2 的账号与密码才可以顺利的登入 pc2 呐！

  这样的架构在目前小型办公室里面是最常见的。例如办公室里面有十个人，每个人桌上

可能都安装有一套 Windows 操作系统的个人计算机，而这十部计算机都可以独立进行办公室软件的执行啊、独立上网啊、独立玩游戏啊等等的，因为这十部计算机都可以独立运作，所以不会有一部计算机关掉，其它的计算机就无法工作的情况发生，这就是 peer/peer 的典型架构。

以下图的架构为例，在这样的架构下，假设 A 君写了一个报告书，而 B 君想要以网络直接取用这个报告书时，他就必须要知道 A 君使用的计算机的账号与密码，并且 A 君必须要在 PC A 上面启用 Windows 的『资源共享(或者是共享)』之后，才能够让 B 君联机进入喔 (此时 PC A 为 Server) ! 而且， A 君可以随时依照自己的喜好来更改自己的账号与密码，而不受 B 君的影响，不过， B 君就得要取得 A 君同意取得新的账号与密码后，才能够登入 PC A 喔! 反过来说，同样的， A 要取得 B 的数据时，同样需要取得 PC B 的账号与密码后，才能够顺利登入啊 (此时 PC A 为 Client 喔)! 因为 PC A, PC B, PC C 的角色与地位都同时可以为 Client 与 Server ，所以就是 peer/peer 的架构了!



生活周遭中，哪里看到的这种 peer/peer 的架构呢? ! 想起来了吗? ! 没错! 就是藉由 Windows 的『网络上的芳邻』所达成的『工作群组(workgroup)』的架构，那就是典型的 peer/peer 架构啦! 所以， peer/peer 也可以直接说成 workgroup 的联机架构喔。

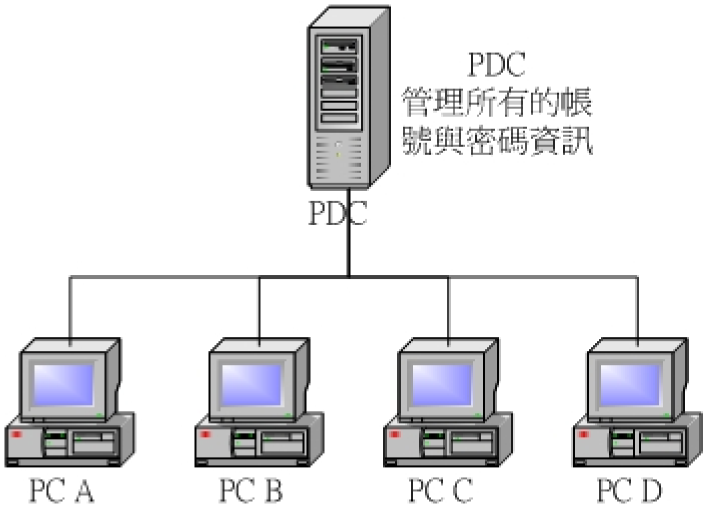
使用 peer/peer 的架构的好处，是每部计算机均可以独立运作，而不受他人的影响! 不过，缺点就是当整个网域内的所有人员都要进行数据分享时，光是知道所有计算机里面的账号与密码，就会很伤脑筋了! 所以， Peer/Peer 的架构是比较适合 (1)小型的网域，或者是 (2)没有需要常常进行档案数据分享的网络环境，或者是 (3)每个使用者都独自拥有该计算机的拥有权(就是说，该计算机是使用者的，而不是公用的啦!)! 而，如果该单位的所有 PC 均是公有的，而且您需要统一控管整个网域里面的账号与密码的话，那就得使用底下的 domain models 了!

- domain model :

假设今天您服务的单位有 10 部计算机，但是您的单位有 20 个员工，这也就是说，这 20 个员工轮流抢着用这 10 部计算机。如果每部计算机都如同 peer/peer 的架构时，那么每部计算机都需要输入这 20 个员工的账号与密码来提供他们登入喔，而且，今天假如有个员工想要变更自己的密码时，就需要到 10 台计算机上面进行密码变更的作业! 否

则他就必须要记得这 10 部计算机里面，那一部计算机是记忆那一个密码.....好烦那～

如果上述是这样的情况，使用 peer/peer 架构就不是一个好方法了！这个时候就需要藉由 domain model 来达成您的需求啦！所谓的 domain model 概念其实也很简单，既然使用计算机资源需要账号与密码，那么我将所有的账号与密码都放置在一部主控计算机(Primary Domain Controller, PDC) 上面，在我的网域里面，任何人想要使用任何计算机时，都需要在屏幕前方输入账号与密码，然后通通藉由 PDC 主机的辨识后，才给予适当的使用权限，也就是说，不同的身份还具有不一样的计算机资源使用权限就是了！例如底下的图示：



PDC 主机控管整个网域里面的各个机器 (PC A ~ PC D) 的账号与密码的信息，假如今天有个使用者账号名称为 Ken ，且密码为 mypasswd 时，他不论使用哪一部计算机 (PC A ~ PC D) 只要在屏幕前方输入 ken 与他的密码，则该机器会先到 PDC 上面查验是否有 ken ，以及 ken 的密码，并且 PDC 主机会给予 ken 这个使用者相关的计算机资源使用权限。当 ken 在任何一部主机上面登入成功后，他就可以使用相关的计算机资源了！

这样的架构比较适合人来人往的企业架构，当系统管理员要控管新进人员的计算机资源使用权时，可以直接针对 PDC 来修改就好了，不需要每一部主机都去修修改改的，对于系统管理员来说，这样的架构在控管账号资源上，当然是比较简单的啦！

各种架构适用的环境与适用的人都不相同，并没有那个是最好啦！请依照您的工作环境来选择联机的模式啰！当然， SAMBA 可以达到上述两种模式的啦！底下我们会分别来介绍喔！

套件安装：

事实上， SAMBA 的安装一点也不难，而且这个咚咚在各主要 distribution 上面都有提供，也都大同小异，所以，比较建议使用您自己的 distribution 所提供的 RPM 档案来安装喔！当然啦，您也可以自行使用 Tarball 来安装的啦！

使用 RPM 来安装

使用 RPM 来安装真是一点都不难啦！不过，要注意一下安装的套件名称就是了，因为不同的 distribution 对于 RPM 档案的命名都不太一样！举个例子来说， Red Hat 9 对于 SAMBA 这个服务器总共需要至少三个套件，分别是：

- samba：这个套件主要包含了 SAMBA 的主要 daemon 档案 (smbd 及 nmbd)、 SAMBA 的文件档 (document)、以及其它与 SAMBA 相关的 logrotate 设定文件及开机预设选项档案等；
- samba-common：这个套件则主要提供了 SAMBA 的主要设定档 (smb.conf) 、 smb.conf 语法检验的测试程序 (testparm)等等；
- samba-client：这个套件则提供了当 Linux 做为 SAMBA Client 端时，所需要的工具指令，例如挂载 SAMBA 档案格式的执行档 smbmount 等等。

不过，在 Mandrake 9.1 当中，则将 samba 这个套件又分为 samba-server 与 samba-doc 两个套件，所以在 MDK 9.1 则有四个套件需要安装： samba-server, samba-doc, samba-common, samba-client 。

RPM 的安装不用再介绍了吧？！请拿出您的光盘， mount 上他，然后再将里头的 samba 套件给他 RPM 上去即可！在最后检验的时候，您的系统应该有点像底下这个样子(以 Red Hat 9 为例)：

```
[root@test root]# rpm -qa | grep samba
samba-common-2.2.7a-8.9.0
redhat-config-samba-1.0.4-1
samba-2.2.7a-8.9.0
samba-client-2.2.7a-8.9.0
注意一下，上面显示的例子是 Red Hat 9 的档案，其中那个 redhat-config-samba
是 Red Hat 额外提供的设定功能，可以不用安装他啦！
```

使用 Tarball 来安装

一般来说，因为各个 distribution 提供的 SAMBA 的功能都差不多，所以实在没有必要使用 Tarball 来进行额外的安装与设定，不过，如果您还是想要自己建置自己的 SAMBA 的话，可以到 SAMBA 的官方网站上下载 samba 的原始程序代码，然后在自己的机器上面编译。不过，连到国外去总是有点慢，建议可以到中山大学下载最新的 SAMBA 原始码：

http://ftp.nsysu.edu.tw/Unix/Samba/

目前 (2003/07/20) 最新的版本是 2.2.8a 这个版本，您可以下载 samba-2.2.8a.tar.gz 这个档案，然后将他解开，解开后，记得察看一下 samba-2.2.8a 里面的 README 喔！接下来，您就

可以这样编译看看：(注：底下的测试是在 Red Hat 9 上面进行的，如果您的系统并不是 Red Hat 9 时，请注意 source 那个目录底下的 config.log 那个档案，里面会记录相关的错误讯息喔！)

```
[root@test root]# wget http://ftp.nsysu.edu.tw/Unix/Samba/samba-2.2.8a.tar.gz
注：上面的网址只是一个范例，请自行到中山大学下载！

[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/samba-2.2.8a.tar.gz
这个时候会有一个目录跑出来： /usr/local/src/samba-2.2.8a

[root@test src]# cd samba-2.2.8a   #(在这个目录中察看一下 README 喔！)
[root@test samba-2.2.8a]# cd source
[root@test source]# ./configure --prefix=/usr/local/samba \
> --with-automount --with-smbmount --with-pam \
> --with-mmap --with-quotas --with-libsmbclient
还是要重复的给他强调一下：
1. 请先以 ./configure --help 察看一下 configure 的一些相关的参数用法
2. 如果发生任何错误，请不要往下进行 make 的动作，因为还是不对的！
3. 万一发生任何错误时，通常是由于一些函式库找不到的缘故，请参考此目录
   下的 config.log 这个档案的内容，里面会记录一些错误的历程。

[root@test source]# make   #(开始进行编译！)
这个过程会花一些时间，因为他会将原始码 (source code) 以您刚刚的设定
并以 gcc 这个 compiler 来进行编译喔！所以会花一些时间的啦！

[root@test source]# make install
将刚刚编译完成的可执行 binary 档案安装到 /usr/local/samba 里面去！
在这个例子当中，未来您在设定 SAMBA 时，必需要到 /usr/local/samba 当中喔！
```

一般来说，除非您的 Linux distribution 已经相当的老旧了 (例如 Red Hat 6.x 以前的版本)，并且在旧的系统上面正在正常的运作一些服务，而仅想要增加 SAMBA 的服务，那就只好使用 Tarball 的方式来安装 SAMBA ，否则的话，蛮强烈的建议直接以 RPM 的方法来安装您的 SAMBA 服务器软件即可！因为既简单方便，又容易统一设定喔！ ^_^

Server 端的设定

由于 SAMBA 几乎一定包含在各个主要的 Linux distribution 当中，并且不同版本之间的功能差异也不是很大，所以，底下的介绍我们都以 RPM 安装的 SAMBA 套件来进行说明。当然啦，即使同样是 RPM 的档案，但是在各个 Linux distribution 当中， SAMBA 的主要档案放置的目录还是可能会不太一样。不过，因为 SAMBA 的设定档档名都是不变的 (smb.conf)，所以，虽然底下我们是以 Red Hat 9 为范例，不过，您依旧可以使用 locate, find, whereis 等指令在不同的 distribution 系统下找出 SAMBA 主要的设定档与执行档喔！ (这就是为什么我们喜欢教大家使用 vi 以及纯文字模式学习 Linux 的原因，因为一法通，万法通啊！)

另外，我一开始的范例当中都是针对没有设定防火墙的情况下所进行设定与测试，如果您的环境

里面已经有架设防火墙的话，那么您应该要先了解防火墙的架构，并将 SAMBA 需要的 port 给他开放，否则很难测试成功喔！或者直接察看本章节较后面专门谈安全的部分，尤其是 iptables 与 /etc/hosts.allow(deny) 这部份喔！

SAMBA 的套件结构

我们这里以 Red Hat 9 的 SAMBA 套件来介绍他相关的一些设定档与执行档，不过，如果您的 distribution 并不是 Red Hat 9 ，那也没有关系，因为都是大同小异的啦！善用 locate 这个指令去搜寻喔！

- SAMBA 的设定档：
  在较早期的版本中， SAMBA 的设定档都直接放置在 /etc 底下，后来的版本则将设定档通通放置到 /etc/samba 底下去了 (有的 distribution 放在 /etc/smb 有的则是 /etc/samba.d ，请使用 locate 搜寻！)。在 /etc/samba 底下的几个重要的设定档有：

  - /etc/samba/smb.conf：这个就是 SAMBA 最主要的设定档了！在较为简单的设定当中，这也是唯一的一个设定档！此外，这个档案本身就含有相当丰富的说明，所以，在设定之前，请使用 vi 好好的详细的观看一下这个档案吧！这个设定档主要的设定分为两部份，分别是 [global] 这个设定主机功能的项目，以及接下来的每个分享出去的目录的属性设定。我们会在后续的部分进行更多的说明。

  - /etc/samba/lmhosts：这个档案的主要目的在对应 NetBIOS name 与该主机名称的 IP ，事实上，他有点像是 /etc/hosts 的功能！只不过这个 lmhosts 对应的主机名称是 NetBIOS name 喔！不要跟 /etc/hosts 搞混了！由于目前 SAMBA 的功能越来越强大，所以通常只要您一启动 SAMBA 时，他就能自己捉到 LAN 里面的相关计算机的 NetBIOS name 对应 IP 的信息，因此，这个档案通常可以不用设定了！

  - /etc/samba/smbpasswd：这个档案预设并不存在啦！他是 SAMBA 预设的使用者密码对应表。当我们设定的 SAMBA 服务器是较为严密的，需要使用者输入账号与密码后才能登入的状态时，使用者的密码预设就是放置在这里咯 (当然啰，您可以自行在 smb.conf 里面设定密码放置的地方及密码文件名，不过，我们这里都以预设的状态来说明) 。比较需要注意的是，这个档案因为包含了使用者的密码，所以，当然权限方面要较为注意啦!这个档案的拥有者需要是 root ，且权限设定为 600 才行喔!

- SAMBA 的执行档：
  SAMBA 的执行档可就多了！一般来说，做为 SAMBA Server 的执行档有 testparm, smbd, nmbd, smbpasswd，至于做为 SAMBA Client 的执行档主要则是：smbmount, smbclient。

  - smbd 与 nmbd：还记得我们在原理部分提到的 SAMBA 需要启动的 daemons 吧？！呵呵！这两个执行档就是那两个主要的 daemons 啰！每次启动 SAMBA 都会使用到的两个执行档啦！

  - testparm：当我们设定完成了 smb.conf 这个主要设定档之后，而想要查看一下 SAMBA 的所有设定参数与 smb.conf 的设定项目是否正确时，就需要使用这个 testparm 来查看啰（ 其实就是 test parameters 的简写！)！所以说，每次在修改完 smb.conf 之后，请务必要使用 testparm 查看看是否有设定错误喔！

  - smbpasswd：如果您的 SAMBA 设定的较为严格，需要规定使用者的账号与密码，那么那个密码档案的建立就需要使用 smbpasswd 来建置才可以的喔！所以这个指令与建立 SAMBA 的密码有关咯！

  - smbclient：当你的 Linux 主机想要藉由『网络上的芳邻』的功能来查看别台计算机所分享出来的目录与装置时，就可以使用 smbclient 来查看啦！这个指令也可以使用在自己的 SAMBA 主机上面，用来查看是否设定成功哩！

  - smbmount：在 Windows 上面我们可以设定『网络磁盘驱动器』来连接到自己的主机上面，同样的，在 Linux 上面，我们可以透过 smbmount 来将远程主机分享的档案与目录挂载到自己的 Linux 主机上面哪！不过，其实我们也可以直接使用 mount 这个指令来进行同样的功能就是了。

- SAMBA 的相关目录：
  这部份需要较为注意的应该算是 SAMBA 的『登录档』吧！因为最近以来，利用『网络上的芳邻』来进行破坏的病毒是越来越多了！而且也有越来越多的搞怪者会以网络上的芳邻的相关漏洞进行入侵的伎俩，所以啰，了解一下登录档放置的地点，并且加以分析，呵呵！可以得到不小的监测呢！

  - /usr/share/doc/samba：这个目录包含了 SAMBA 的所有相关的技术手册喔！也就是说，当您安装好了 SAMBA 之后，您的系统里面就已经含有相当丰富而完整的 SAMBA 使用手册了！值得高兴吧！ ^_^，所以，赶紧自行参考喔！

- /var/log/samba：这个目录就是 SAMBA 预设的登录文件放置目录了！如果您的 SAMBA 老是设定不起来，又或者怀疑被人家以 port 137~139 入侵的话，就到这里来观察吧！

- /usr/share/samba/codepages：这个目录里面放置的就是各个语言的支持格式。举例来说，想让您的 SAMBA 支持中文吗？那么就需要 codepage.950 这个档案的支持啰！当然啦，在 smb.conf 里面设定即可！

主机的规划技巧建议

如果您的 Linux 主机单纯要用来做为档案服务器的话 (File Server) ，那么建议您 Linux 主机就不要安装 X Window 的咚咚，以节省一些硬盘的空间。此外，如果您想要针对不同的使用者开放不同的登入权限，那表示您的 SAMBA 主机将会有很多人物同时进进出出的存取数据文件。为了避免某些使用者占用了大部分的硬盘空间，也为了维护上的便利，这个时候挺建议您将 /home 这个目录独立出一个 parition ，此外，空间也要大一点，因为每个使用者登入 SAMBA 系统的时候，预设都是会进入到个人的家目录的，而 Linux 预设的个人家目录就在 /home 底下，所以呐，/home 是需要大一些些的。

- 在安装 Linux 的时候，建议不需要安装 X Window ；
- 在规划 Linux 时，/home 最好独立出一个 partition ，而且硬盘空间最好能够大一些；
- /home 独立出来的 partition 可以单独进行 quota 的作业，以规范每个使用者能够使用的最大硬盘容量；
- 由于 SAMBA 可以做为打印机服务器，所以建议打印机可以直接连接在 Linux 主机的打印端口 (LPT1)；
- 由于 SAMBA 一般来说都仅针对内部（LAN）主机进行开放，所以，可能的话， SAMBA 主机直接使用内部保留 IP 来设定即可 (Private IP)，当然啦， SAMBA 是否使用 private IP 还得视您的整个网域的 IP 网段的特性来规划。以我们研究室来说，因为实验室所有计算机的 IP 都是 Public IP，那么 SAMBA 如果使用 Private IP ，当然大家都无法连接上啊！ ^_^
- 如果您的 SAMBA 主机使用 Public IP 时，请特别留意规范好防火墙的设定，尽量仅让 LAN 内的计算机可以联机进来即可，不要对 Internet 开放喔！

基础的设定流程与 smb.conf 的主要规划

在开始设定 SAMBA 这个服务器之前，我们先来谈一谈应该如何较为简单的设定 SAMBA 吧！因为 SAMBA 的功能很强大，可以做为简单的单一主机控管自己分享出去的资料，也可以做为整个区域内所有计算机的账号管理主机（ Primary Domain Controller, 这部份我们会在后面进行介绍喔！）。不过，整个 SAMBA 的设定流程倒是没有多大的差异性！嗯！既然 SAMBA 是要让 Linux 加

入 Windows 网络上的芳邻的一项工具，那么我们就先来谈一谈，在 Windows 上面，您要如何分享你机器上面的目录给大家使用呢？

10. 先在自己的计算机上面安装必要的协议：那就是 NetBIOS（ 有时候会是 NetBEUI 喔！）的安装咯！直接在网络设定里面设定好即可；
11. 再来则是在档案总管里面设定好要分享的目录、磁盘或者是装置(如打印机)；
12. 然后，给这个分享出来的咚咚一组账号及密码(如果需要的话)，让外部使用者可以使用这组账号密码登入 Windows 主机；
13. 然后就开始运作了！

整个流程大概就是这么简单吧！事实上，在 Linux 底下的设定也是这么简单的啦！

14. 先参考 局域网络的设定 那一章节，先搞定硬件的联机吧！
15. 之后，先在 linux 上面的 SAMBA 设定档 smb.conf 里面设定好主机所支持的各项功能，例如是否需要密码、是否支持 PDC 等等；
16. 然后在 smb.conf 的后半部当中设定好想要分享的目录与该目录的属性；
17. 如果在步骤 2 里面的 smb.conf 档案内设定的分享方法是需要账号与密码的登入时，就以 smbpasswd 建立使用者的账号与密码；
18. 启动 smb 的服务，开始运转啰！

呵呵！所以会动到的设定档几乎就是只有 smb.conf 这个 SAMBA 的设定档，当然啦，还需要 smbpasswd 来建立使用者的账号与密码就是了！ ^_^既然这个 smb.conf 这么重要，我们就得了解一下 smb.conf 啰！

smb.conf 这个档案里面主要是以 [global/share directory] 开始一个主要设定的内容，这个档案里面，『 # 』与『 ; 』都是批注的意思喔！我们先来谈一谈比较重要的 [global] 这个牵涉到 SAMBA 主机的主要设定的内容吧！

```
[root@test root]# cp /etc/samba/smb.conf /etc/samba/smb.conf.bak.raw
[root@test root]# vi /etc/samba/smb.conf
# 这个档案本身就是很详细的说明档，限于篇幅，我们没有将批注的部分写下来，
# 这里的目的是想让大家了解 smb.conf 的内容规划咯！
[global]
   workgroup = birdhouse
   server string = Linux Samba Server
   netbios name  = birdlinux
   client code page = 950
   printcap name = /etc/printcap
   load printers = yes
   printing = lprng
```

```
   log file = /var/log/samba/log.%m
   max log size = 500
#  那个 500 数字是 Kb 喔!
   security = user
   encrypt passwords = yes
   smb passwd file = /etc/samba/smbpasswd
   socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
# 上面的几行当中，注意 [global] ，由 [global] 到底下的 [Webpage] 之间的设定
# 都是 [global] 的设定项目！而 [global] 的主要内容都是与主机的设定有关!
# 比较重要的有底下几个设定值：
# workgroup     工作群组：同一个局域网络内，要具有相同的 workgroup
# netbios name  主机名称：这个主机名称就是 netbios 的名字！请注意，如果你
#               没有设定 netbios name 的话，预设的 NetBIOS name 会以
#               HOSTNAME 来替代!
# server string 这个是主机的说明，随便写写没关系!
# security      这个重要，是用来规定 SAMBA 主机的安全登入项目，有底下几种：
#               share  ：不进行安全登入，亦即没有设定账号与密码
#               user   ：设定主机的密码文件作为登入的验证档案，这与底下的
#                        smb passwd file 有关喔!
#               domain ：就是让您的 SAMBA 作为 PDC 啰!
# log file      登录文件放置的目录所在喔!
# 其它相关的几个设定项目请参考 man 5 smb.conf 我们底下也会列出几个常用的
# 设定项目说～

[Webpage]
   comment      = My Home Page
   path         = /var/www/html
   read only    = no
   public       = yes
   writable = yes
   create mode = 0664
   directory mode = 0775
# 这个部分则是针对每个分享的目录或者是装置进行权限方面的规定了!
# 几个简单的设定项目有：
# comment  ：这个目录的说明!
# path     ：这个项目真正的 Linux 档案系统里面的目录，请看底下的说明
# read only：是否只读？
# public  ：是否让所有可以登入的使用者看到这个项目？
# writable ：是否可以写入？！这里需要注意一下喔！那个 read only 与 writable
#            不是两个蛮相似的设定值吗？如果 writable 在这里设定为 no ，亦即
#            不可写入，那跟 read only 不就互相抵触了？！那个才是正确的设定？
#            答案是：最后出现的那个设定值为主要的设定!
# create mode 与 directory mode 都与权限有关的咯!
```

注意一下，在上面的案例中，我们只有两个主要的设定群：

- 一个是 [global] 的主机相关设定，这个是每个 SAMBA 主机都需要设定的喔！里面的规定都与 SAMBA 主机的环境有关！

- 至于 [Webpage] 呢？他是什么玩意～这里我们举个实际的例子好了，当您在局域网络内以网络上的芳邻登入某部主机之后，不是会看到该主机所提供的『目录或者装置的名称』吗？！那个 Webpage 就是名称啦！也就是说，当您以网络上的芳邻登入 linux 时，看到的 Linux 所分享出来的目录名称就是『Webpage』啦！不过，这个 Webpage 只是在 SAMBA 服务当中所显示的名称而已，并不是真正的 Linux 档案系统上面的目录！这个 Webpage 所代表的真正的目录要看底下设定项目的『path』设定项目才行！这也就是说，当您在网络上的芳邻登入 Linux 后，看到 Webpage 这个目录，用鼠标将他点下去，接下来看到的内容就是 /var/www/html 这个真正 Linux 档案系统里面的数据咯！

- 关于变数：
  在上面的例子当中，我们有看到一个比较有趣的设定是『log file = /var/log/samba/log.%m』，怎么会有个 %m 啊？！呵呵！那个就是 SAMBA 里面的变量值啦！在 SAMBA 当中，为了便利使用者的设定，所以会有许多的变量值提供给系统管理员来使用，主要的变量有底下几个：

  - %S：取代目前的设定项目值，所谓的『设定项目值』就是在 [ ] 里面的内容！举例来说：

```
[homes]
  valid users = %S
  ....
```

  - 因为 valid users 是允许的登入者，设定为 %S 表示任何可登入的使用者都能够登入的意思～今天如果 test 这个使用者登入之后，那个 [homes] 就会自动的变成了 [test] 了！这样可以明白了吗？！ %S 的用意就是在替换掉目前 [ ] 里面的内容啦！

  - %m：代表 Client 端的 NetBIOS 主机名称喔！例如上面案例的登录档！
  - %M：代表 Client 端的 Internet 主机名称喔！就是 HOSTNAME。
  - %L：代表 SAMBA 主机的 NetBIOS 主机名称。

  - %H：代表使用者的家目录。

- %U：代表目前登入的使用者的使用者名称

- %g：代表登入的使用者的群组名称。

- %h：代表目前这部 SAMBA 主机的 HOSTNAME 喔！注意是 hostname 不是 NetBIOS name 喔！

- %I：代表 Client 的 IP 咯。

- %T：代表目前的日期与时间

至于相关的变量运用，我们会在底下的设定当中略做介绍喔！

没有防备的 SAMBA 分享档案设定

所谓的『没有防备的 SAMBA 分享档案』就是你启用了一个 SAMBA Server ，设定了分享的目录，但是却完全没有规范权限，也就是任何人都可以登入这个系统的意思啦！事实上是不太应该介绍这个没有防备的 SAMBA 主机的！因为.....太过于危险了！不过，有些没有连上 Internet 的局域网络内还是可以尝试看看的。所以这里我们先以较为简单的无防备 SAMBA 主机作为第一个 SAMBA 的设定介绍吧！

11. 主机预计分享的状况：
    在整个 LAN 里面的工作群组 (workgroup)为：birdhouse
    我的 Linux 主机 NetBIOS 名称为 (netbios name)：birdhome
    安全设定为没有防备的 share (share 为 smb.conf 里面 security 的设定值)
    仅仅分享 /tmp 这个目录而已～

12. 设定 lmhosts ：
    事实上，这个档案目前是可以不用设定了！不过，如果保险一点来看，设定一下也没有什么不好的!在这个档案当中,您要设定的数据很简单,就是每一部 PC 的 NetBIOS name 以及对应的 IP 即可！以我为例：

```
[root@test root]# vi /etc/samba/lmhosts
127.0.0.1      localhost
192.168.0.100  birdhome
192.168.0.110  birdbrother1
192.168.0.120  birdbrother2
```

13. 在我的区网当中总共有三部计算机，分别是 Linux 主机的 192.168.0.100 以及两部 Windows 主机 192.168.0.110 及 120 ，请注意喔！这三部计算机的工作群组都必须是 birdhouse 而计算机名称请个别取 birdbrother1, birdbrother2 等不同的主机名称呢！再次强调，这个名称是 NetBIOS name 喔！

14. 开始设定 smb.conf：
    在这个例子当中，我们仅分享出 /tmp 而已，并且没有设定任何的登入权限的限制喔！而因为 smb.conf 原本的设定当中就已经开放出很多的目录，所以您必须要将其它的分享先关闭！关闭的方法有很多，您可以将 smb.conf 备份后移除，重新建一个，或者是将没有用到的那一行以『;』或者是『#』将他批注掉喔！(比较详细的说明请参考前几个章节的『基础设定流程』里面的介绍！)

```
[root@test root]# cd /etc/samba
[root@test samba]# cp smb.conf smb.conf.bak
# 玩服务器最重要的一个概念就是『有备无患』啊！
# 所以，先将重要的数据给他备份下来！ ^_^

[root@test samsa]# vi smb.conf
# 底下的设定为最基础的设定值！最重要的地方在于 security = share 的地方！
[global]
   workgroup = birdhouse
   netbios name = birdhome
   server string = Bird's testing SAMBA Server
   client code page = 950
#  这个 client code page 的设定有趣的很！因为 SAMBA 支持多语系的编码，
#  我们习惯的编码为 cp590 亦即是 code page 950 这个编码，所以，
#  想要让您的 SAMBA 可以正确的在 Windows 上面显示出中文，就得加入
#  client code page = 950 喔！ (如果没有设定，那么默认值是 850 呢！)
#  如果 max log size = 0 的话，那表示登录档档案大小没有限制！
   log file = /var/log/samba/log.%m
   max log size = 0
   security = share
   socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
   interfaces = 192.168.0.100
   dns proxy = no
[tmp]
   comment = Temporary file space
```

```
   path = /tmp
#  真正的开放出来的路径在这里！
   read only = no
   public = yes
#  上面两个设定在告诉大家，不但可以存取(read only = no)
#  也可以让大家查询(public = yes)!

[root@test samba]# grep -v '^#' smb.conf |grep -v '^;'|grep -v '^$'
# 这个动作在确认一下上面的设定是否相同，因为有时候可能会忘记将某个
# 设定给批注掉呢！ ^_^
```

15.
16. 测试 smb.conf 设定值与启动 SAMBA ：
    设定好了最主要的设定档 smb.conf 之后，接下来就得要开始测试与启动 SAMBA 啰！动作也是很简单的：

```
[root@test samba]# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[tmp]"
Loaded services file OK.
Press enter to see a dump of your service definitions  <==这里按下 Enter
# Global parameters
[global]
        client code page = 950
        code page directory = /usr/share/samba/codepages
        workgroup = birdhouse
        netbios name = birdhome
        netbios aliases =
        netbios scope =
        server string = Bird's test SAMBA Server
        interfaces = 192.168.0.100
        bind interfaces only = No
        security = SHARE
        encrypt passwords = No
.....
[tmp]
        comment = Temporary file space
        path = /tmp
        read only = No
        guest ok = Yes
# 你应该会看到如上的画面，这个 testparm 可以用来察看所有 SAMBA 的属性，
# 这些属性很多都是默认值！如果没有 smb.conf 里面设定时，就是默认值，
```

```
[root@test samba]# /etc/rc.d/init.d/smb restart
Shutting down SMB services:                              [  OK  ]
Shutting down NMB services:                              [  OK  ]
Starting SMB services:                                   [  OK  ]
Starting NMB services:                                   [  OK  ]
# 请注意，上面 /etc/rc.d/init.d/smb 这个档名是 Red Hat 的设定值，
# 很多其它的套件不见得是使用这个档名的！例如 OpenLinux 使用 samba 这个，
# 所以请您务必使用 RPM 的相关指令来检验一下文件名称，或者使用
# <tab> 按键来让系统自动补齐文件名，就可以知道是什么档名啰！
# 另外，如前所说的， SAMBA 会启用两个 daemons ，所以这里显示两个 OK!

[root@test samba]# netstat -tlunp | grep ':13'
tcp    0   0 0.0.0.0:139          0.0.0.0:*         LISTEN      4307/smbd
udp    0   0 192.168.0.100:137    0.0.0.0:*                     4311/nmbd
udp    0   0 0.0.0.0:137          0.0.0.0:*                     4311/nmbd
udp    0   0 192.168.0.100:138    0.0.0.0:*                     4311/nmbd
udp    0   0 0.0.0.0:138          0.0.0.0:*                     4311/nmbd
# 如果看到这样，呵呵！就应该没有问题啦！
```

17. 事实上，当您完成了 smb.conf 的设定时，请务必使用 testparm 来检查一下 smb.conf 的设定是否正确喔！因为很多时候，我们都会一不小心忘记这、忘记那的，所以，使用 testparm 来查阅一下设定值，确实有帮助的呐!此外，启动后，记得去察看一下 port 是否有启动喔！还有还有，不要忘记了，如果您原先就有设定防火墙的话，一定要去察看一下防火墙的设定是否已经启动了 137 ~ 139 的登入？！

18. Client 端的测试：
    Client 端的测试在 Windows 或 Linux 当中都差不多，所以我将他独立到另一个章节来说明，底下仅列出在本机上面的自我测试(亦即本机是主机，同时也是 client 就是了！)

```
[root@test root]# smbclient -L //birdhome
added interface ip=192.168.0.100 bcast=192.168.0.255 nmask=255.255.255.0
Password:  <==这里按 Enter
Domain=[birdhouse] OS=[Unix] Server=[Samba 2.2.7a-security-rollup-fix]

        Sharename      Type      Comment
        ---------      ----      -------
       tmp          Disk     Temporary file space
        IPC$           IPC       IPC Service (Bird's testing SAMBA Server)
        ADMIN$         Disk      IPC Service (Bird's testing SAMBA Server)

        Server              Comment
        ---------           -------
       birdhome          Bird's testing SAMBA Server

```

```
    Workgroup            Master
    ---------            -------
   birdhouse           birdhome
```

19. 看到上面显示的吗？!那个 smbclient 指令可以用来(1)查询某部主机的分享内容与(2)登入某部主机进行数据的存取！更详的用法我们会在底下说明！在上面的案例下，我们使用 -L 这个参数，去察看主机名称(注意，这里是以 netbios name 为主机名称喔！)分享出来的数据有哪些？!如上所示，列出 birdhome 该部主机的分享的目录 (tmp) 以及主机名称还有群组特性！这就表示 SAMBA 设定完成啦！

这样就简单的设定完毕了！大家都可以使用网络上的芳邻登入您的 Linux 主机，并且使用 /tmp 这个目录喔！而，如果您想要增加其它的目录开放给大家使用时，就请自行模仿 [tmp] 底下的设定值，

不过，需要特别留意的仅是 Linux 的档案权限与 SAMBA 设定的权限关系！这个问题我们会在安全性与问题克服里面进行详细的说明喔！(注:这个问题最常发生在使用者身上,因为『即使 SAMBA 主机设定您可以无限制的使用某个目录下的档案，但是是否能够使用，仍然得视登入 SAMBA 的该使用者对于 Linux 的档案系统是否有存取的权限』喔！)

设定需要使用者登入的 Workgroup

上面介绍了没有防备的 SAMBA 主机之后，您是否觉得：『呵呵！SAMBA 还真是不错用』啊！是没错啦！设定方面确实很简单，然而还是有缺点的，就是万一有外人不小心在网络上的芳邻上面点一点、按一按，刚好进入到您的主机系统当中，由于您的主机系统是没有防备的，所以他可以自由的进出您的主机，也可以随意的将数据下载(或者是上传)到您的 SAMBA 主机上面，更可能由于局域网络内有 Nimda 等网络芳邻攻击型病毒，而将病毒硬塞一份到您的 SAMBA 主机当中，使得您的局域网络里面的网络频宽被吃掉之外，还可能使其它的局域网络内的 Windows PC 操作系统被搞破坏掉～哇！真是问题多多啊！所以啰！不建议在公共的场合底下设定无防备的 SAMBA ，即使是私人单位内部，还是不建议架设上面的没有任何防备措施的 SAMBA 主机喔！

所以呢,接下来,我们要介绍的就是需要使用者提供账号密码才能登入 SAMBA 主机的设定方法咯！使用者必须要能够提供账号与密码供主机判定身份，若身份合格，才能够使用主机的相关资源喔！底下我们要介绍的是比较简单的 peer/peer 的联机 (相关的联机模式，请参考前面的联机模式一节)，此外，使用的还是 workgroup 的方式来设定的喔！

20. 主机预计分享的状况：
    在整个 LAN 里面的工作群组 (workgroup)为：birdhouse
    我的 Linux 主机 NetBIOS 名称为 (netbios name)：birdhome
    安全设定为工作群组类型 user

分享家目录与特定目录 /home/public 给所有使用者使用

21. 设定 lmhosts：
    同样的，我有三部主机，请参考上一章节的设定值：

22. 开始设定 smb.conf 并检验 smb.conf 的设定参数：
    在这个案例当中，我们要分享的数据有(1)每个人的家目录；(2) /home/public 这个特定目录。我们知道 smb.conf 有个 path 来指定给 SAMBA 真正取用的目录，但是在这个案例中，我们要指定的是每个人的家目录呐！怎么设定好家目录呢？呵呵！就使用变量来给他设定好啊！例如底下的说明：

```
[root@test samsa]# vi smb.conf
[global]

#  底下为一般设定项目(主机名称、工作群组等)
   workgroup = birdhouse
   netbios name = birdhome
   server string = Bird's testing SAMBA Server
   client code page = 950

#  与安全有关的登入信息项目，这个 security = user 一定要设定，
#  而密码需加密，此外，密码档案放置在 /etc/samba/smbpasswd 里面，
#  这个档案需要自行设定起来喔！等一下会介绍！
   security = user
   encrypt passwords = yes
   smb passwd file = /etc/samba/smbpasswd

#  一些与安全性有关的设定，其中， hosts allow 较为有趣一点！
#  当您设定了 hosts allow 之后，未在 hosts allow 里面规定的 IP
#  将无法登入 Linux 的 SAMBA 主机喔！特别留意呐！如果您想要完全开放 IP，
#  或者使用防火墙管理，那就不要设定 hosts allow 了！
   hosts allow = 192.168.0.  127.
   log file = /var/log/samba/%m.log
   max log size = 0
   socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
   interfaces = 192.168.0.100
   dns proxy = no

# 底下则设定每个使用者的家目录！
[homes]
   comment = Home Directories
   browseable = no
   writable = yes
   valid users = %S
```

```
   create mode = 0664
   directory mode = 0775
#  create mode 指的是新建立档案的权限，而 directory mode 则是新建目录的权限
#  至于 valid users 则是有权限进入者，这里设定为 %S 喔！所以，
#  每个使用者就可以进入自己的家目录了！

[public]
   comment = Public Stuff
   path = /home/public
   public = yes
   writable = yes

[root@test samba]# grep -v '^#' smb.conf |grep -v '^;'|grep -v '^$'
# 这个动作在确认一下上面的设定是否相同，因为有时候可能会忘记将某个
# 设定给批注掉呢！ ^_^

[root@test samba]# mkdir -p /home/public
# 上面既然有提到要开放 /home/public ，当然，这个目录就必须要存在啊！
```

23. 其实重点只有在于 security = user 以及 encrypt passwords 与 smb passwd file 这三行！有了这三行，您的 SAMBA 就需要去查询使用者要登入时的账号与密码了。同时，请注意在 [homes] 里面的设定当中，有一个 valid users 对吧！这个 valid users 指的是:『能够登入这个目录的使用者是谁？』的意思!那么谁才有权力能够登入您的 SAMBA 服务器呢？！呵呵！这个底下直接告诉你。

24. 设定使用者账号与密码：
设定使用者账号是很重要的一环，因为设定错误的话，当然也就任何人都没有办法登入的！在这里我们必须先要说明一下 Linux 的档案系统与 SAMBA 设定的使用者登入权限的相关系！

    - 在 Linux 这个系统下，任何程序都需要取得 UID 与 GID (User ID 与 Group ID) 的身份之后，才能够拥有该身份的权限，也才能够适当的进行存取档案等动作！
    - 关于 Linux 这个系统的 UID 与 GID 与账号的相对关系，通通记录在 /etc/passwd 这个档案当中；
    - SAMBA 仅只是 Linux 底下的一套软件，使用 SAMBA 来进行 Linux 档案系统时，还是需要以 Linux 系统下的 UID 与 GID 为准则！

如果上面这几点说明您没有问题了，现在就来看一下当我们在 Windows 计算机上面以网络上的芳邻来连接 Linux 并且进行数据的存取时，会是怎样的一个情况呢？

- 我们需要透过 SAMBA 所提供的功能来进行 Linux 的存取，而 Linux 的存取是需要取得 Linux 系统上面的 UID 与 GID 的，因此，我们登入 SAMBA 主机时，所利用 SAMBA 取得的其实是 Linux 系统里面的相关账号！这也就是说，在 SAMBA 上面的使用者账号，必须要是 Linux 账号中的一个！

如果上面您可以理解了，那么就可以知道底下的这个基本规则：『在 SAMBA 主机所提供能够登入的账号，必须要在 /etc/passwd 里面存在！』，也就是说，如果您想要使用 ken 登入 SAMBA 主机，那么在 Linux 上面就必须要存在有 ken 这个账号，如此一来，当您以 ken 登入 SAMBA 主机时， SAMBA 才能够去 /etc/passwd 找到相对应的 UID 与 GID ，来提供您登入 SAMBA 之后取得的程序的相关权限！这个咚咚相当的重要，如果这里搞不清楚，在后面的一些目录权限的设定就会通通搞乱掉了！

所以说，如果您需要以 bird 这个账号登入 SAMBA 时，并且 Linux 本身并没有 bird 这个使用者，呵呵！那么您就必须要使用 useradd 来使 Linux 系统多出一个名为 bird 的账号，然后才可以让该账号登入 SAMBA 服务器喔！并且，并不是所有在 /etc/passwd 里面的账号都可以用来登入 SAMBA 主机，必须要使用 SAMBA 的相关功能 (就是 smbpasswd 这个指令) 所新增到 SAMBA 密码设定文件里面的账号才可以使用 SAMBA 登入喔！废话不多说，来假设一个例子吧！假设 Linux 已经具有 bird, bigbird, smallbird 三个账号，而我只想要让 bird 使用 SAMBA 而已，其它两个账号不想开放，那么我就只要这么做即可：

```
0. 根据 smb.conf 的设定，建立一个密码文件！
   如果您是第一次建立 SAMBA 的使用者，才需要进行这个动作：
   因为我们在 smb.conf 里面设定密码 smb passwd file = /etc/samba/smbpasswd
[root@test root]# cd /etc/samba
[root@test samba]# touch smbpasswd
[root@test samba]# chown root:root smbpasswd; chmod 600 smbpasswd
# 请注意， smbpasswd 这个档案记录了能够使用 SAMBA 服务器的使用者账号
# 与密码，所以当然只有 root 才能够进行读写了！特别留意其权限啊！

1. 开始建立密码：
[root@test samba]# smbpasswd -a bird
New SMB password: <==在这里输入 bird 的密码
Retype new SMB password: <==再输入一遍 bird 的密码
Added user test.
# 请特别留意， SAMBA 的密码是放在 /etc/samba/smbpasswd 这个档案内，当然，
# 您可以更改这个档名 (在 smb.conf 里面改)，但是， Linux 系统的账号密码
# 是放在 /etc/shadow ，这也就是说， SAMBA 服务器的密码与 Linux 底下的
# 账号密码并不一定要相同的！至于 smbpasswd 的使用大致上有几个参数：

2. smbpasswd 之语法解释
语法：smbpasswd [-adem] username
参数：
```

```
   : 如果都没有加上任何一个参数，亦即『smbpasswd bird』时，这表示：
     修改 SAMBA 密码文件(/etc/samba/smbpasswd)里面的 bird 这个账号的密码！
     也就是说，密码文件里面已经存在一个 bird 的账号了！
-a : 在 smbpasswd 密码文件里面新增一个使用者
-d : 让在 smbpasswd 密码文件里面的某个账号的使用者暂时无法使用 SAMBA
     当多了 -d 的参数时，在 smbpasswd 里面某个字段会多出一个 D 的参数，
     代表该账号目前无法使用喔！
-e : 与 -d 参数相反，让某个账号恢复使用！
-m : 该 username 为机器代码(Machine Account)，这个与 domain model 有关！
范例：
[root@test samba]#  more smbpasswd
bird:1001:01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A82807973B89537:[UX          ]:LCT-3F
[root@vbird samba]# smbpasswd -d bird
Disabled user bird.
[root@vbird samba]# more smbpasswd
bird:1001:01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A82807973B89537:[DUX         ]:LCT-3F
# 特别留意，当使用 -d 之后，该账号会在密码档案里面的中括号之特殊字段中，
# 多出一个 D 的参数，让该账号暂时无法使用喔！
```

在上面的例子当中，我们仅有开放 bird 才能登入我们的主机喔！此外，在 smb.conf 里面有个设定 valid users ，这个 valid users 所规定可以使用 SAMBA 的账号，就是在 /etc/samba/smbpasswd 里面出现的账号啰！

25. 重新启动 SAMBA 服务器与 Client 端的测试：
    重新启动 SAMBA 很简单啊，还是使用 /etc/rc.d/init.d/smb 来动作，至于 client 端的测试就需要其它的要求了！

```
[root@test samba]# testparm
[root@test samba]# /etc/rc.d/init.d/smb restart
[root@test samba]# smbclient -L //127.0.0.1
added interface ip=192.168.0.100 bcast=192.168.0.255 nmask=255.255.255.0
Password: <==直接按 Enter
Anonymousloginsuccessful
Domain=[BIRDHOUSE] OS=[Unix] Server=[Samba 2.2.7a-security-rollup-fix]
        Sharename      Type      Comment
        ---------      ----      -------
        public         Disk      Pulic Stuff
        IPC$           IPC       IPC Service (Bird's testing SAMBA Server)
        ADMIN$         Disk      IPC Service (Bird's testing SAMBA Server)
...(略)...
```

```
# 注意看上面，因为我们没有在 smbclient 指定登入者，所以登入
# 后的身份会被指名为匿名者 anonymous 喔！由于我们有开放 public 给大家
# 浏览，所以还是看的到咚咚的！(注：我有省略一些输出！)

[root@test samba]# smbclient -L //127.0.0.1 -U bird
added interface ip=192.168.0.100 bcast=192.168.0.255 nmask=255.255.255.0
Password: <==这里请务必输入正确的 bird 的 SAMBA 密码!
Domain=[BIRDHOUSE] OS=[Unix] Server=[Samba 2.2.7a-security-rollup-fix]
        Sharename      Type      Comment
        ---------      ----      -------
        public         Disk      Pulic Stuff
        IPC$           IPC       IPC Service (Bird's testing SAMBA Server)
        ADMIN$         Disk      IPC Service (Bird's testing SAMBA Server)
  bird           Disk      Home Directories
...(略)...

# 仔细分辨一下上下两个不同点。在多加了 -U username (-U bird 那个地方)
# 由于登入者的身份变成 bird 了，因此我们就可以看到 bird 的家目录了！
# 也就是粗体字那一行啊！这样可以清楚的知道了？！
```

26.
27. 关于权限的简略说明：
    由前面在介绍 smbpasswd 时，您大概已经可以知道 SAMBA 账号(在 /etc/samba/smbpasswd 里面)与 Linux 账号(在 /etc/passwd 里面)的差异咯，若当我们以 SAMBA 登入 Linux 主机后，会取得一个使用者的相关权限，此外，SAMBA 也有自订的权限(writable, read only, public 等等的参数，均会影响登入者的权限喔！)，这些权限的相关性为何？！反正有个大前提一定要知道的，无论您使用任何 process 在 Linux 上面，该程序都需要符合 Linux 系统的权限概念，也就是说， Linux 本身的档案权限大于 SAMBA 对于使用者所设定的权限！

    举上面的例子来说好了，那个 /home/public 我们在 SAMBA 中设定的是 writable 喔！所以，当我以 bird 登入 SAMBA 服务器后，对于 /home/public 应该是具有可以读写的能力的！但是，偏偏刚刚我是以 root 的身份来建立 /home/public ，因此该目录仅有 root 可以写入(权限为 755)，因此， bird 是无法在 /home/public 底下进行写入的动作的！所以，当我以网络上的芳邻并以 bird 登入 Linux 的 SAMBA 服务器，结果想要将数据传输到 /home/public ，屏幕就会显示『您没有权限写入』之类的字眼～不要怀疑，绝对就是『Linux 权限的问题』啦！因此，这个时候请利用你的 chown 或者 chmod 指令来修改一下该目录的权限吧！重要重要喔！

上面的设定案例应该蛮足以提供一般家庭用的环境中进行设定了！如果您还要扩充分享的目录与能够登入的使用者，可以这样做：

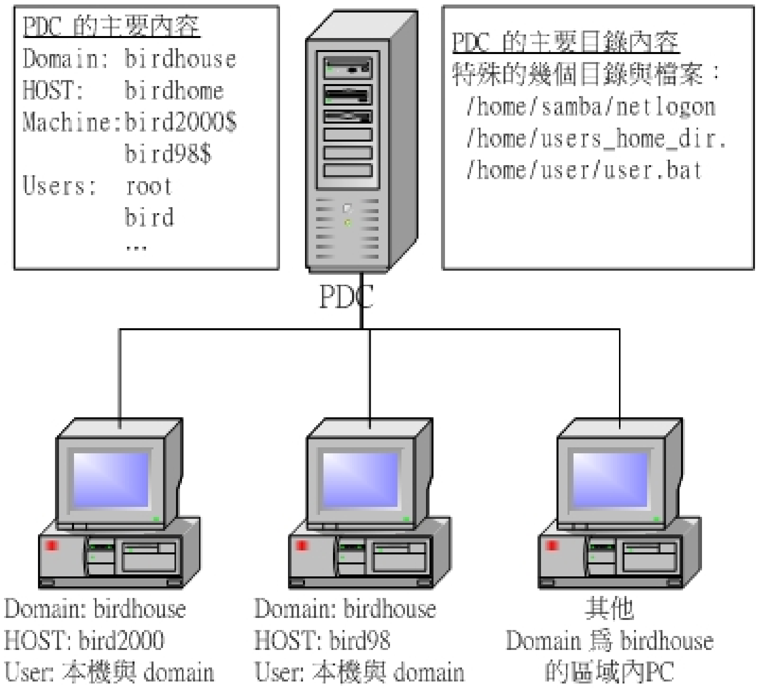
- 利用编辑 smb.conf 来多开放其它的目录，并且特别注意 Linux 在该目录下的权限喔！请使用 chown 与 chmod 吧！
- 利用 smbpasswd 来新增其它使用者到 /etc/samba/smbpasswd 里面去，如果该账号并没有出现在 /etc/passwd 里面，请先以 useradd 新增该账号；
- 不论进行完任何的设定，请先以 testparm 进行确认，之后以 /etc/rc.d/init.d/smb restart 来重新启动！

事实上，SAMBA 的一般用途就是在这个联机的模式中！多使用 SAMBA 来分享您的资源吧！我都是使用 SAMBA 来做为远程服务器与我的工作机互通有无的重要媒介说～

设定较大型网域的 Linux Samba PDC (Primary Domain Controller)主机

上面介绍了两种方法，比较常用的还是那个需要使用者登入信息的工作群组方法，该方法相当的适合小型的网域呢！但是，如果是比较大型的网域，那就比较伤脑筋了～就如同在原理部分我们提到的联机方法里面的 domain model 的模式(请参考原理部分的说明)。一般来说，如果在比较大型的网域系统当中，通常系统管理员都会选择 domain model 的联机模式，如此一来在使用者账号与密码以及登入后取得的计算机资源控管上显的相当的容易。不过，这种 domain model 比较常见于 Windows NT 的架构当中，也就是说，控管整个 domain 里面的使用者账号与密码，其实是由 Windows NT 服务器来管理的呐！所以说，现在如果我们想要使用 SAMBA 主机作为整个 domain 里面的的一员，并且 SAMBA 主机本身并不管理自己的账号与密码，那么 SAMBA 自然就得要将账号与密码的查询转给另一部提供账号、密码查询的主机来服务与辨识身份啦！当然啰，如果您的网域当中并没有 Windows NT 的 PDC 主机，那么使用 SAMBA 作为 PDC 主机，以提供整个网域的账号与密码的验证工作,也是一个可行的方案!底下我们要介绍的,就是以 SAMBA 当作 PDC 的一个简易的设定！请留意喔，我们底下说明的是一个简单的 PDC 主机，如果您还需要额外的磁盘分享，请发挥您自己的创意喔！(修改 smb.conf)

31. 预计达成的架构：



就如同上面的图示，我们在 birdhouse 这个网域当中，连同 PDC 这部 birdhome 总共有四部计算机，整个 birdhouse 的网域是由 birdhome 这部 Linux SAMBA 所设定成的 PDC 来掌控的，这部 PDC 需要哪些数据呢？

- 需要有每一部在这个网域内的 PC 的机器代号 (Machine account)，这个 Machine account 就是各个主机的 NetBIOS 名称，也就是 bird2000, bird98 等主机名称。不过，请特别留意，在各个 PC 上面的主机名称为 bird2000 与 bird98，但是在 PDC 的账号名称上面，需要加上 $ 在后端，也就是在 PDC 上面需要设定 machine account 成为 bird2000$, 与 bird98$ 才行！

- 由于 Windows NT (例如 Windows 2000 这个操作系统)第一次登入 PDC 时，需要使用系统管理员的身份建立联机，所以， PDC 需要设定 root 这个使用者账号才行！此外， PDC 会去搜寻每部 PC 要登入 PDC 时，他的 machine account ，所以，在 /etc/passwd 与 /etc/sabma/smbpasswd 里面需要同时具有：(1)每个 machine account 的账号； (2)root 这个账号需要加入 smbpasswd 里面； (3)任何想要使用 PDC 登入的该网域的使用者账号，例如我们这个案例的 bird 这个使用者。所以啰，在这个案例中， /etc/passwd 与 /etc/samba/smbpasswd 需要有底下的账号：bird2000$, bird98$, bird, root 至少要有这些账号喔！如果还需要让 bird2, bird3... 等人登入的话，就需要使用 useradd 及 smbpasswd 新增使用者啰！(注:特别特别注意,那个 root 的账号只有在您的网域当中有 Windows 2000 或者是 Windows XP 的操作系统 时

才需要加入 /etc/samba/smbpasswd 里面！这是因为第一次登入 PDC 时，需要以 root 的身份来设定好总总的联机才行！)

- 在 SAMBA 这部 PDC 上面需要开放的目录有哪些呢？
  /home/samba/netlogon：当使用者在其它 PC 登入 PDC 的身份认证时，PDC 会依据使用者的设定文件(profile)来配置给这位登入者相关目录与权限的开放，这个目录就是在放置使用者的设定档了！设定档的档名通常为 username.bat ，不过我这里设定每个人都使用类似的 profile ，所以档名统一为 startup.bat 喔！请注意，这个档案必须是 DOS 的档案！
  /home/user：使用者的家目录啊！
  /home/samba/profiles：每个使用者在登入 SAMBA 所模拟的 PDC 之后，还可以取得『自己的 Windows 设定值』喔！这包含了『我的最爱、每个 Windows 软件的相关设定、通讯簿....一大堆资料』。都可以在这个地方给他写入呢！

如果一切设定都没有问题，那么当使用者在 bird2000 这部个人计算机以网域的型态登入时， PDC 会主动的依据 /home/samba/startup.bat 里面的使用者设定数据，配置『网络磁盘驱动器』到 bird2000 那部计算机上面去，以及将 /home/profiles/user 的个人设定值加载到 bird2000 去！所以 bird 这个使用者就可以在 bird2000 这部计算机上面使用 SAMBA 上头的数据了！而且使用的桌面设定啦、我的最爱啦等等的数据都是 SAMBA 上面的喔!同样的,未来如果 bird 这个使用者在 bird98 这部计算机登入到 PDC 时，仍然会得到 bird 个人的网络磁盘驱动器 (就是 /home/bird)，啊哈！那么 bird 不论在哪里都可以自由自在的使用自己在 Linux SAMBA 服务器上面的资料啦！ ^_^

32. 设定 smb.conf 设定档：
    我们预计分享的目录与权限就如同上一个步骤的说明，不过，我还想要额外的分享出 /tmp 这个目录就是了！那么整个 smb.conf 的简易设定可以是这样的！

```
[root@test samsa]# vi smb.conf
[global]
#1.底下为一般设定项目(主机名称、工作群组等)
   workgroup = birdhouse
   netbios name = birdhome
   server string = Bird's testing SAMBA Server
   client code page = 950
#2.密码与登录文件相关的信息!
   security = user
   encrypt passwords = yes
   smb passwd file = /etc/samba/smbpasswd
   log file = /var/log/samba/%m.log
   max log size = 0
```

```
   socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
#3.其它与 domain 有关的参数
   os level = 64
   preferred master = yes
   domain master    = yes
   local master     = yes
   domain logons    = yes
   logon script     = startup.bat
;   logon script     = %U.bat
   logon path       = \\%L\Profiles\%U
   wins support     = yes
   dns proxy        = no
   time server      = yes
# 在上面的设定当中，都是与 domain model 有关的参数
# os level      表示与其它主机相比，这部 SAMBA 机器的管理等级，设高一点
# domain logons 表示这部 SAMBA 主机可以提供 Widnows 登入的服务 (PDC)
# logon script  表示当使用者登入之后，要到哪里去执行他的 profile 设定档，
#               我这里将每个使用者登入时，都要去执行 startup.bat 这个档案，
#               这个档案放置的地方其实就是底下 [netlogon] 的 path
#               设定的目录，请特别留意 domain 与 netlogon 的关系！

# 底下则设定每个使用者的家目录！
[homes]
   comment = Home Directories
   browseable = no
   writable = yes
   valid users = %S
   create mode = 0664
   directory mode = 0775

# 有特定账号者可以使用底下这个设定
[netlogon]
   comment = Network Logon Service
   path = /home/samba/netlogon
   writable = no
   write list = bird root
   follow symlinks = yes
   guest ok = yes

# 没有特定账号者，例如 Windows 98 的使用者，会使用底下的路径！
[Profiles]
    path = /home/samba/profiles
    read only = no
    create mask = 0600
```

```
    directory mask = 0700
    browseable = no

[tmp]
   comment = Temporary file space
   path = /tmp
   read only = no
   public = yes

[root@test samba]# grep -v '^#' smb.conf |grep -v '^;'|grep -v '^$'
# 这个动作在确认一下上面的设定是否相同，因为有时候可能会忘记将某个
# 设定给批注掉呢！ ^_^

[root@test samba]# testparm | more
[root@test samba]# /etc/rc.d/init.d/smb restart
```

33. 在这边的设定当中，最重要的地方在于 [global] 里面的 logon script 以及 [netlogon] 的设定了！这两个设定请特别留意不要设定错误了！否则会真的很麻烦～在这个案例当中，我让每个使用者登入 PDC 辨识身份后，都去执行 /home/samba/netlogon/startup.bat 这个档案就是了！与刚刚我们上面建议的每个使用者使用不同的 scripts 是不一样的设定值喔！请稍微留意一下呐！好了，在这样的设定值之后，我们就可以直接重新启动 smb 了！并且也可以提供 Windows 来登入我们的 PDC 啰！

34. 处理 PDC 主机所需要的各个目录与档案：
    刚刚上面我们设定了很多的不同的数据目录，在这里我们得要好好的给他建立起来！

```
[root@test root]# mkdir /home/samba
[root@test root]# mkdir /home/samba/netlogon
[root@test root]# mkdir /home/samba/profiles
[root@test root]# cd /home/samba/netlogon
[root@test netlogon]# vi startup.tmp
net time \\birdhome /set /yes
net use U: \\birdhome\homes
net use T: \\birdhome\tmp
# net 是 Windows NT 网域的相关指令用法：
# net time : 表示目前 (client) 使用的时间要与 \\server (在这里是 birdhome)
#            同步的意思！有时候因为 Server 与 Client 不同步，某些程序
#            会有问题！
# net use [device:] [directory]
#      device:   那个 device 是 Windows 的磁盘槽啦！
#      directory 是 SAMBA 主机相对的目录！
#      在我这个案例当中，每个使用者登入之后，他会在档案总管当中，
```

```
#       发现有个 U 槽，而且内容是 \\birdhome\homes 相同；
#       发现有个 T 槽，内容则是 /tmp 喔！

# 特别留意， startup.bat 必须要是 DOS 的格式，所以在 Linux 上编辑时，
# 还要加上一些特殊的转换动作！
[root@test netlogon]# cat -A startup.tmp | tr '$' '\r' > startup.bat
[root@test netlogon]# cat -A startup.bat
net time \\birdhome /set /yes^M$
net use U: \\birdhome\homes^M$
net use T: \\birdhome\tmp^M$
# 看到了吗？！每一行的最后面要加上有 ^M 这个 Windows 的杰作才可以！

# 除此之外，我们还要将原先在 bird2000 上的 bird 这个使用的个人设定值
# 给他复制过来 SAMBA 主机上面喔！在 Windows 2000 预设的情况下， bird
# 这个人的设定值会是在：
C:\Documents and Settings\bird
# 请将这个 bird 目录完整的给他复制到 /home/profiles/bird 当中，也就是说，
# 在 SAMBA 主机内的 /home/profiles/bird 里面就有原先 Windows 2000 内的
开始
Application Data
Favorites
My Documents
....
# 等等的档案数据喔！然后这样做：
[root@test netlogon]# cd /home/samba/profiles
[root@test profiles]# chown bird -R bird/
# 这个时候您应该已经将原先 Windows 的 bird 这个人的设定值给他复制到
# /home/samba/profiles/bird 当中了才对！
```

35. 在上面我们建立了 SAMBA 所需要的目录，尤其是那个 netlogon ！然后，还需要建立每个使用者登入都会去读取的 startup.bat 那个档案，注意啊！那个档案是 Windows 的格式，所以如果您在 Linux 上面编辑的话，不要忘记了加上特殊动作来转换格式！当然啦！如果您是在 Windows 的系统上面编辑 startup.bat 的话，那么特殊动作就不需要进行了！此外，由于我们想要让 bird 这个使用者『不论在哪一部 Windows 2000 的机器上面，都可以套用同一组的个人设定值』，这个时候就得要让 /home/samba/profiles 内部存有与使用者账号相同的目录了，也就是 /home/samba/profiles/bird 这个目录喔！在这个目录下就具有 bird 的我的最爱啦、我的活页夹啦等等的资料！因为是 bird 的啊！所以记得要将该目录的所有人设定为 bird 才行喔！(注：/home/samba/profiles/user 每个使用者的设定值最好不要太大，鸟哥曾经测试过，以我的 Windows 2000 原本的设定值完整的给他复制到 SAMBA 主机上，结果竟然发现有 300MB 这么大，导致我每次在 Windows 2000 登入 PDC 主机来取得我的 Profile 都要等大约 10 分钟左右，因为要将 profile 完整的读过来啊！(要传 300MB 的资料量啊！)

36. 设定 Machine account 与 user account ：
    重要的工作来啦！我们要为 PDC 建立与 Client 主机的相关性！也就是在刚刚上面第一个步骤当中，我们提到的 Client 的机器代码 (Machine account)，在我们的案例当中，需要多了个 bird2000$ 与 bird98$ 两个机器代码，您必须要在 /etc/passwd 与 /etc/samba/smbpasswd 里面同时增加这两个机器代码才行！除此之外，因为 bird2000$ 与 bird98$ 其实只是要给 SAMBA 用的账号而已，所以我们不需要给他家目录与 Shell 啊！而且，在 /etc/samba/smbpasswd 当中，还要告诉 SAMBA 这两个账号是机器代码喔！所以您应该这样做：

```
[root@test root]# groupadd smbmachine
# VBird 有时候会有点龟毛，既然我的 /etc/passwd 里面会有 SAMBA 用的机器码，
# 那么我就将这几个机器代码归类在特定的 group 之内，那就自订一个名为
# smbmachine 的 Machine group 好了！ ^_^
[root@test root]# useradd -g smbmachine -d /dev/null -s /bin/false bird2000$
[root@test root]# useradd -g smbmachine -d /dev/null -s /bin/false bird98$
[root@test root]# smbpasswd -a -m bird2000$
[root@test root]# smbpasswd -a -m bird98$
# 注意啊！多了一个 -m 的参数，这个参数代表后面接的为 Machine account ，
# 而不是一般设定的 User account 喔！
[root@test root]# usradd bird
[root@test root]# smbpasswd -a bird
# 假设 bird 这个使用者还没有被建立，那么您应该这样建立他！
[root@test root]# smbpasswd -a root
# 记得要将 root 的身份设进去 smbpasswd 里面喔！
# 不过，只要在 Windows 2000 的登入设定完成之后，就可以将 root 取消掉了！
```

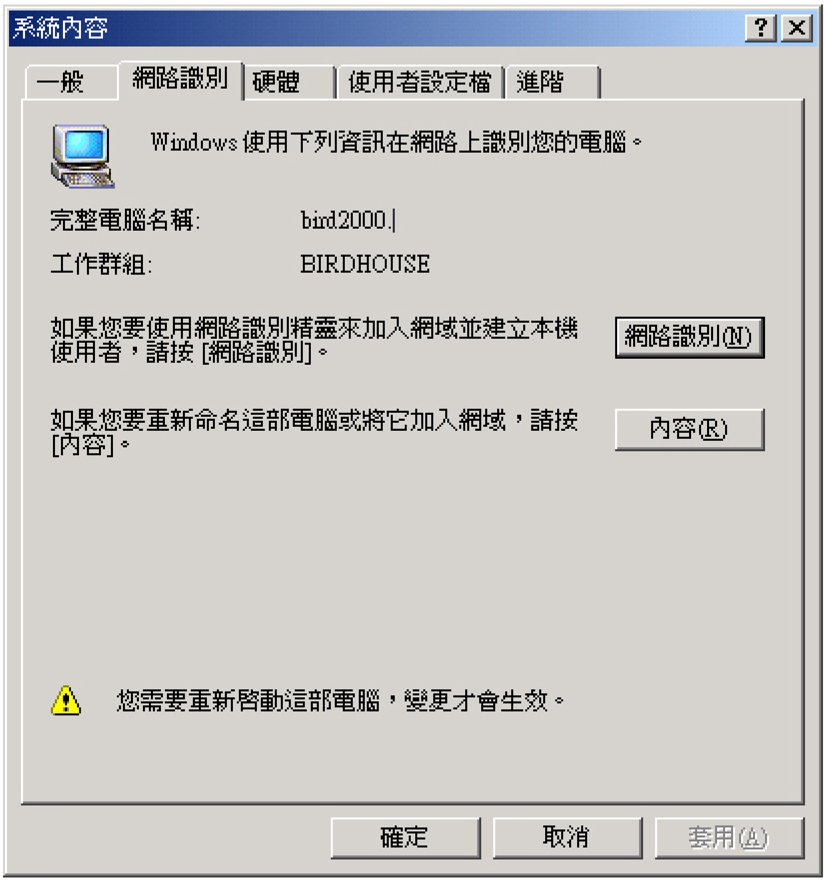
37. 特别的再给他注意喔！因为 bird2000$ 与 bird98$ 只是 machine account ，所以不需要提供 Linux 的密码啊！也就是不需要 passwd bird2000$ 喔！不要去更动 /etc/shadow 的意思啦！ ^_^

38. Client 端登入 PDC ：Windows 2000
    要使用 Windows 2000 登入 SAMBA 做成的 PDC 比较麻烦一点～因为首先我们必须要『先以 root 登入 SAMBA 的 PDC 主机，设定好联机之后，才算成功』的啦！所以说，我们的 /etc/samba/smbpasswd 里面才需要那个 root 使用者！要用 Windows 2000 登入 Linux 的步骤是这样的：

- 『开始』=>『设定』=>『控制台』，在出现的窗口中选择双击『网络和拨号联机』，出现如下窗口：

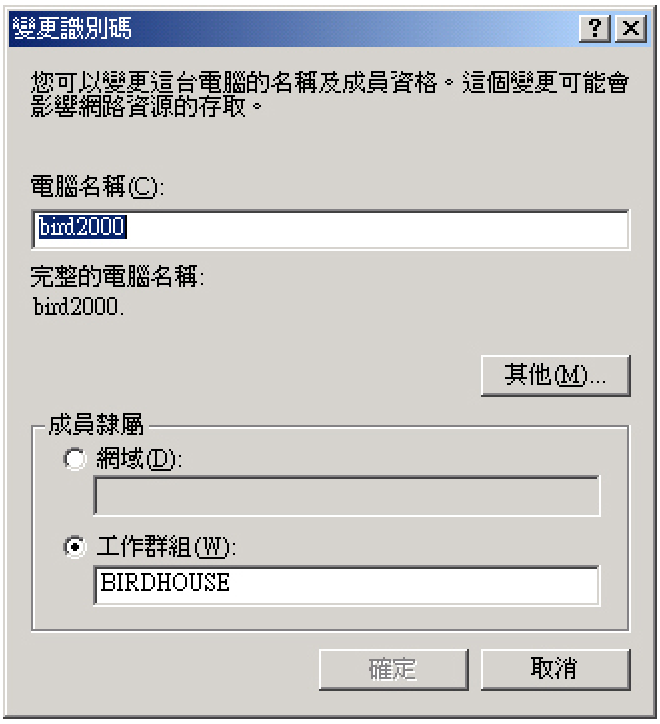


- 在上面的窗口当中，选择『网络识别』，之后出现底下的窗口喔：

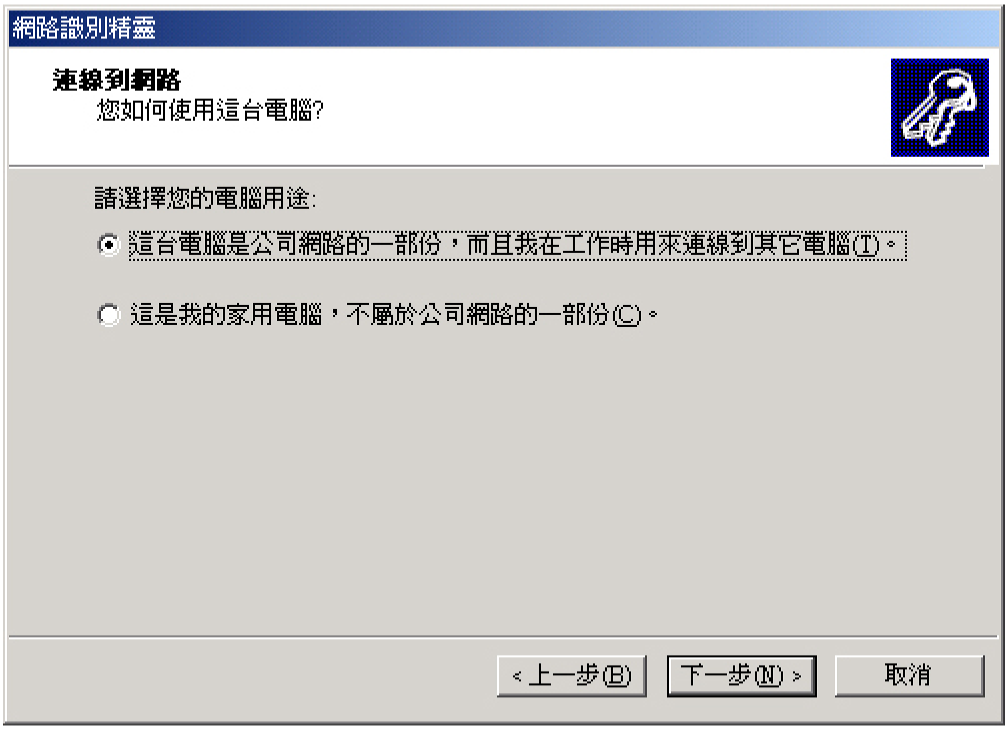


- 由于我们在这个设定中，需要的主机名称为 bird2000 而网域为 birdhouse ，所以，万一您的 Windows 2000 原本设定并不是这样的网络识别状态，那么请选择上面图示的『内容』在出现的窗口当中填入正确的主机名称与工作群组！

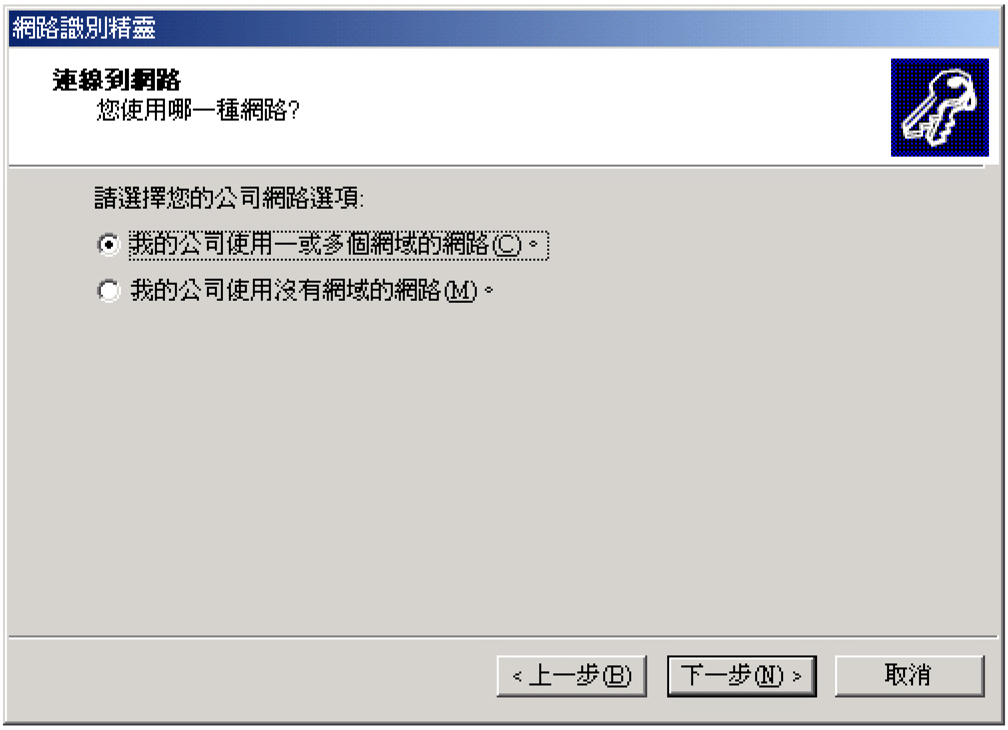
选择『确定』然后重新开机吧！



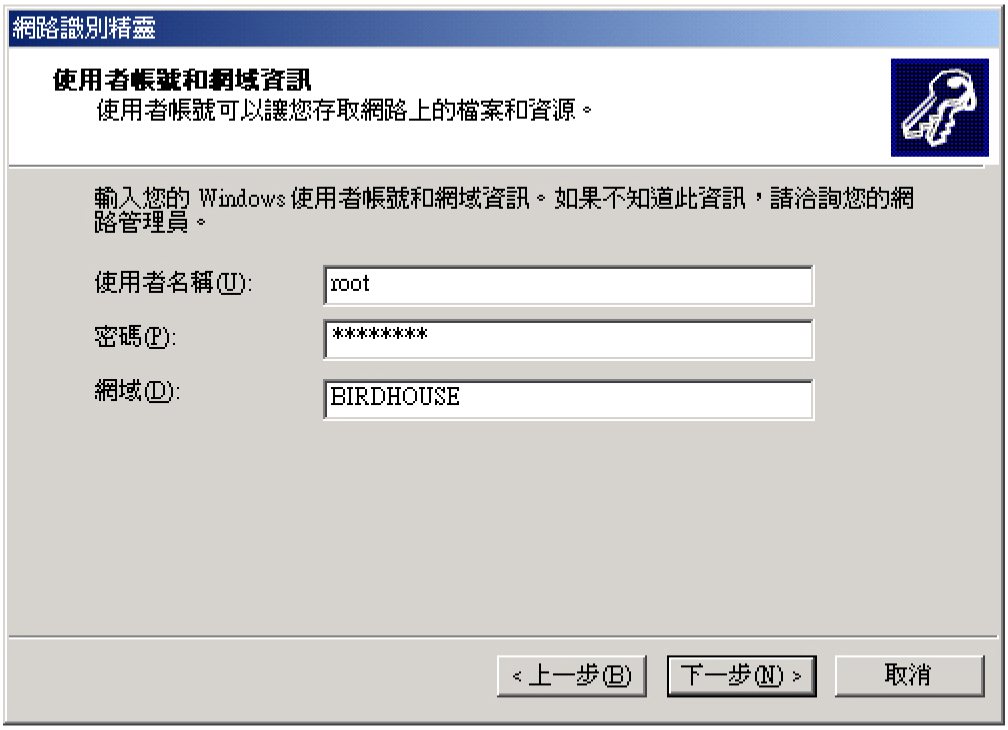
- 重新开机完成之后，还是回到『网络识别』(第二个步骤，也就是上上个图示)的地方，按下『网络识别』，会出现欢迎画面，在按下『下一步』之后，会出现如下的图示：



- 上面的图标请选择『这台计算机是公司网络的一部分』才行！然后选择『下一步』出现如下画面：

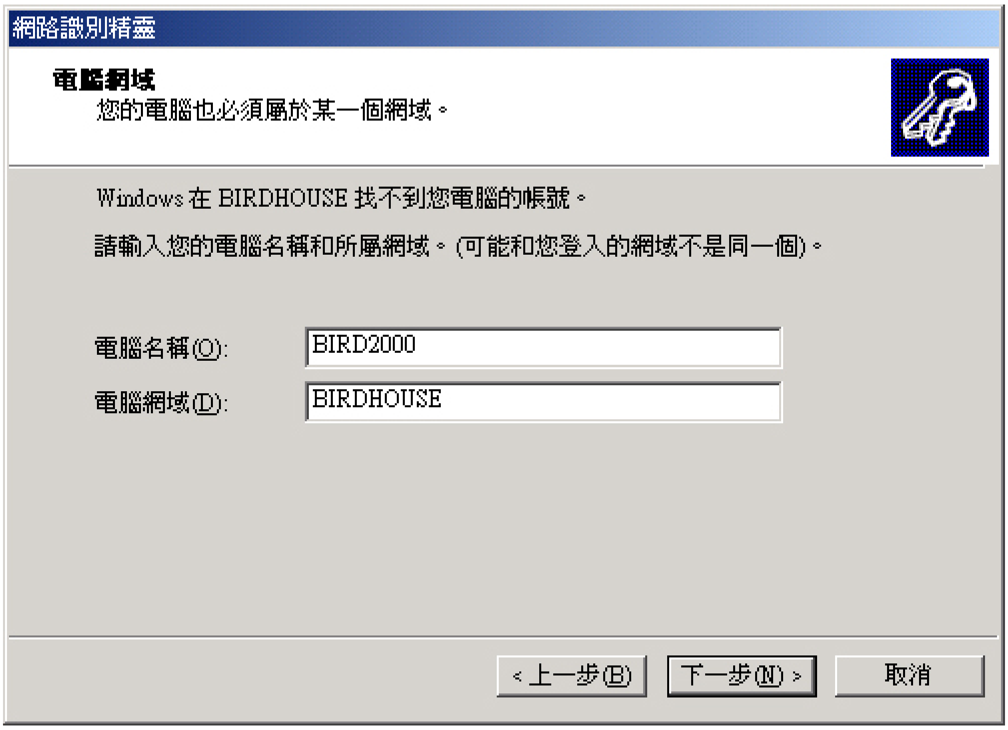


- 上面的图示请选择『我的公司使用一个或多个网域的网络』才行！然后按下下一步，会出现一个警告的讯息，告诉您，您必须要有的数据，这包含了计算机名称、网域使用者名称等等！在按下下一步之后，会出现如下画面：



- 这里请注意啊！第一次登入时，需要在使用者名称的地方输入 root 才行喔！那个『密码』是由 smbpasswd 设定的密码，并不是 root 在 Linux 系统

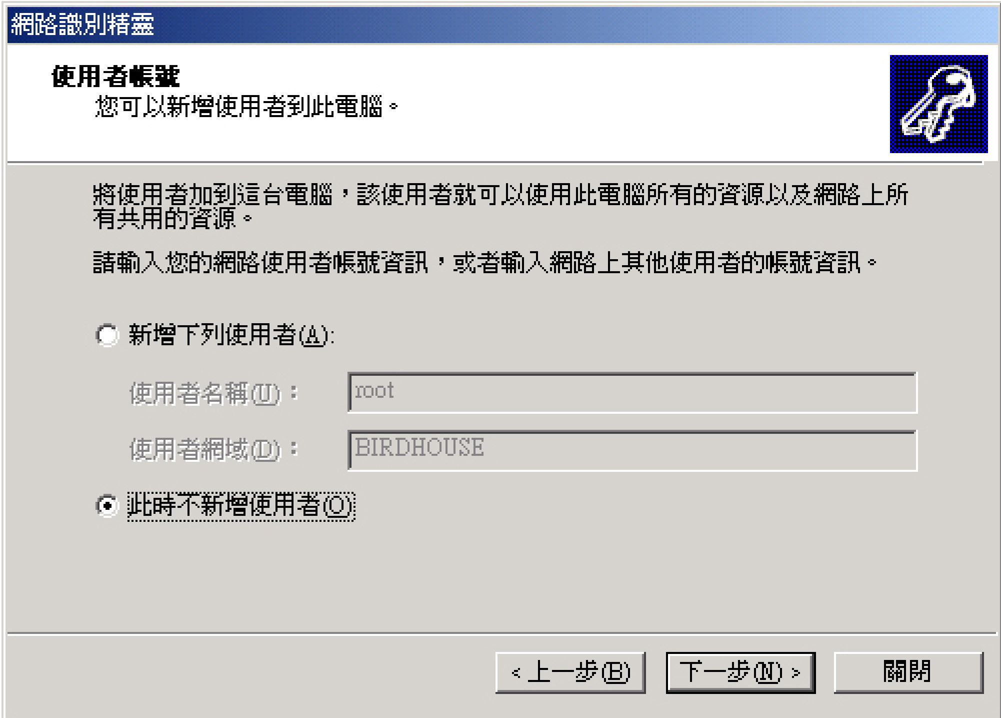
(/etc/shadow) 里面的密码喔！不要搞错了！按下下一步吧！



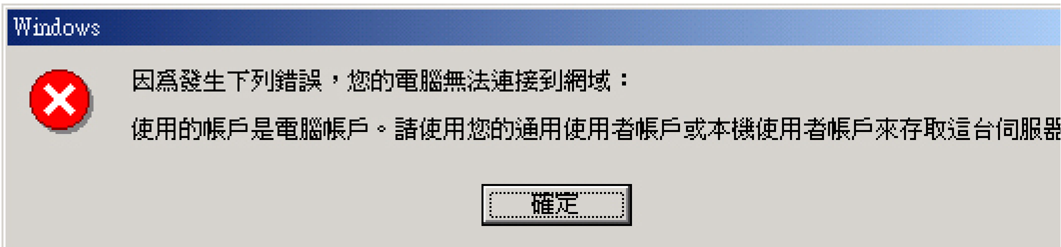
- 由于可能会出现一些小问题，所以这个画面会重复的给他出现的啦！重新输入一下我们的网域与主机名称吧！按下下一步：



- 再次的输入 root 与 root 的密码 (/etc/samba/smbpasswd 里面的密码数据！)，网域填写正确喔！然后按下确定：



- 看到这个画面时，您应该要很高兴的啦！ ^_^！因为这就是正确的显示已经登入了 PDC 主机啦！咦！怎么会有新增使用者呢？！是的，因为 SAMBA 可以提供一个功能 (在 smb.conf 里面设定 add user script 的项目，请参考 man 5 smb.conf) 来让第一次登入 PDC 的 Windows 2000 可以使用 SAMBA 提供的功能来使 Linux 主机增加使用者(同时新增在 /etc/passwd 与 /etc/samba/smbpasswd)！不过，我不太建议这样做啦！因为要新增使用者，只要网管人员登入 SAMBA 主机，使用 useradd 与 smbpasswd 即可，多了这个功能，觉得有点危险就是了 @_@！所以，通常我就直接按下『此时不新增使用者』，呵呵！这样就已经是 OK 啦！
- 万一在您的设定过程当中，老是出现底下的画面：



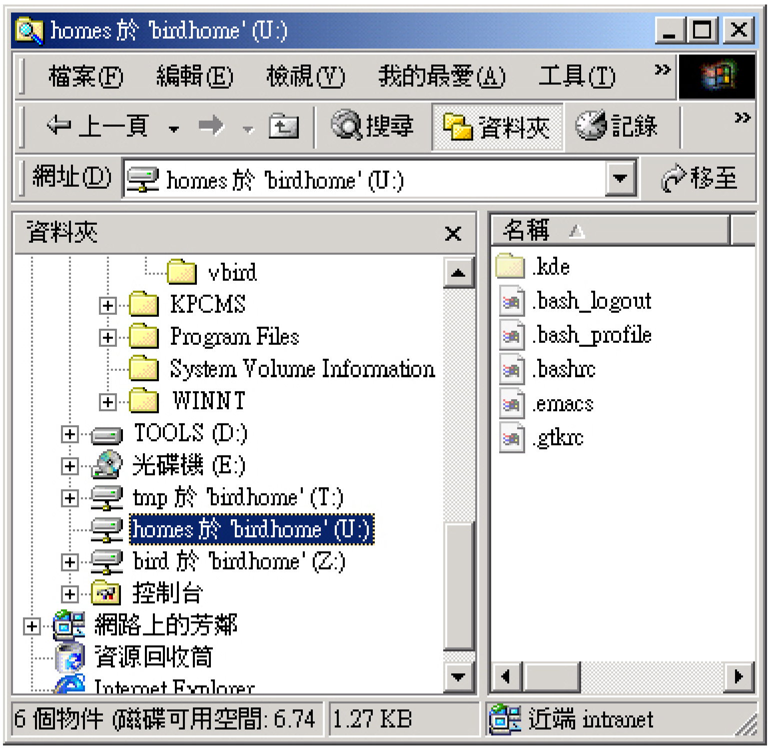
真是太可恶了！果真如此的话，那么建议您可以这样做看看：

1. 先察看一下 /var/log/samba 里面的登录文件信息，尤其是 bird2000.log 是关于 bird2000 这部主机的信息呐！
2. 如果还是无法解决，可以在 lmhosts 里面增加 bird2000 的 IP 与主机名称的对应，然后将 samba 整个关掉『/etc/rc.d/init.d/smb stop』，等待一段时间让 NetBIOS 的名称解析时间逾时，再重新启动

samba『/etc/rc.d/init.d/smb start』, 然后再重新做一次输入 root 的密码那个动作！

在我尝试过的案例中，上面第二个步骤挺有效的！不过，还是得要察看 /var/log/samba 里面的登录信息才行喔！

- 当您再次的将 Windows 2000 重新开机之后，屏幕出现的登录画面会提醒您需要按下『Ctrl-Alt-Del』才能出现登录窗口，并且在登录窗口中会出现三个空格（如果只有两个空格时，请按下『选项』喔，其它的功能就会自动的出现了！）：
    1. 使用者名称：这个名称可以(1)填入本机 Windows 2000 的使用者名称，也可以 (2)填入 PDC 上面的使用者名称，例如 bird ；
    2. 密码：这个密码也要对应上面输入的使用者，看是本机或者是 PDC 主机的使用者相对应的密码；
    3. 登入到：这里可以选择您所需要登入的是本机还是 PDC 网域呐！也需要与上面两个项目对应说！
- 假设我以 bird 登入，并且选择 BIRDHOUSE 网域 来登入之后，开启档案总管，会得到如下的网络磁盘驱动器喔！



所以，我就可以自由自在的使用我自己的网络磁盘啦！不论在那个计算机上面，都可以使用同一个 SAMBA 机器上的目录，并且使用的都是同一组密码（因为记录在 PDC 主机上面），修改密码也很简单啊！ ^_^

经过上面的设定之后，如此一来，您就可以让使用者在各个 Widnows 2000 的环境当中使用您的网络磁盘驱动器啰！ ^_^

39. Client 端登入 PDC ：Windows 98

在 Windows 98 上面登入 PDC 要比 Windows 2000 来的简单一些些说～整个作法可以这样来：

- 先清除桌面，然后在桌面将鼠标指针移动到『网络上的芳邻』上头，按右键，选内容，会出现如下画面：

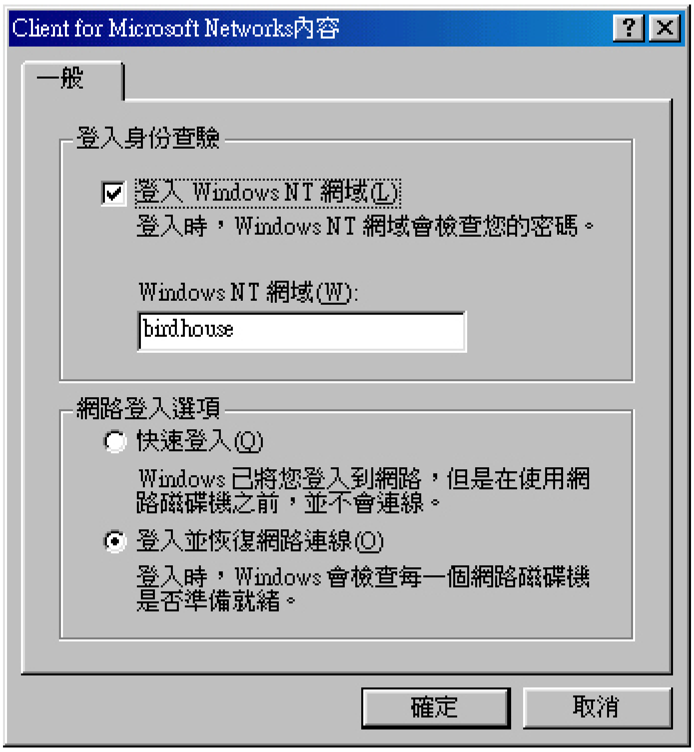


务必确认在上面的窗口当中，最上头的空白部分含有：

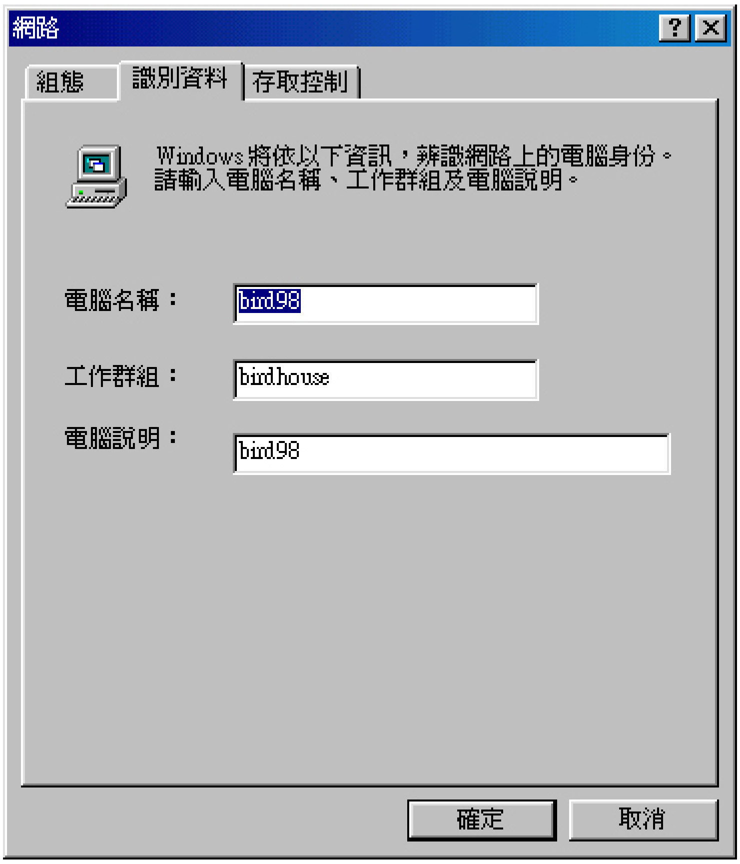
1. TCP/IP 的协定：这个用来设定你的 IP 以及网络参数
2. NetBEUI 的协定

然后特别留意喔！在底下的空格上面，必须要是『Client for Microsoft Networks』的主网络登录模式，之后，请在上方的空格处，双击『Client for

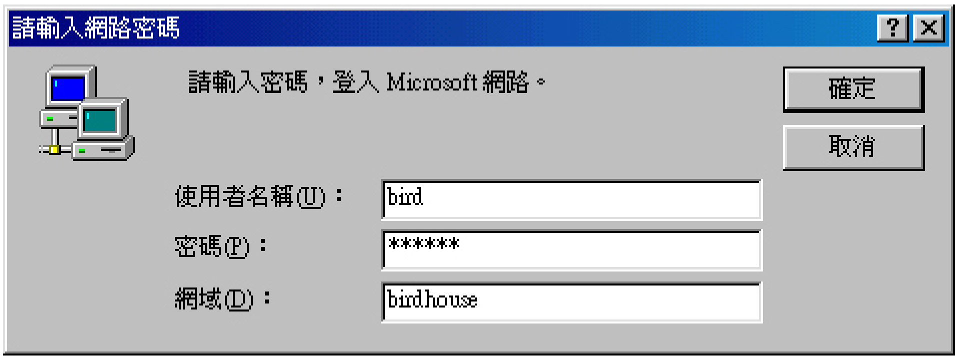
Microsoft Networks』，会出现如下画面：



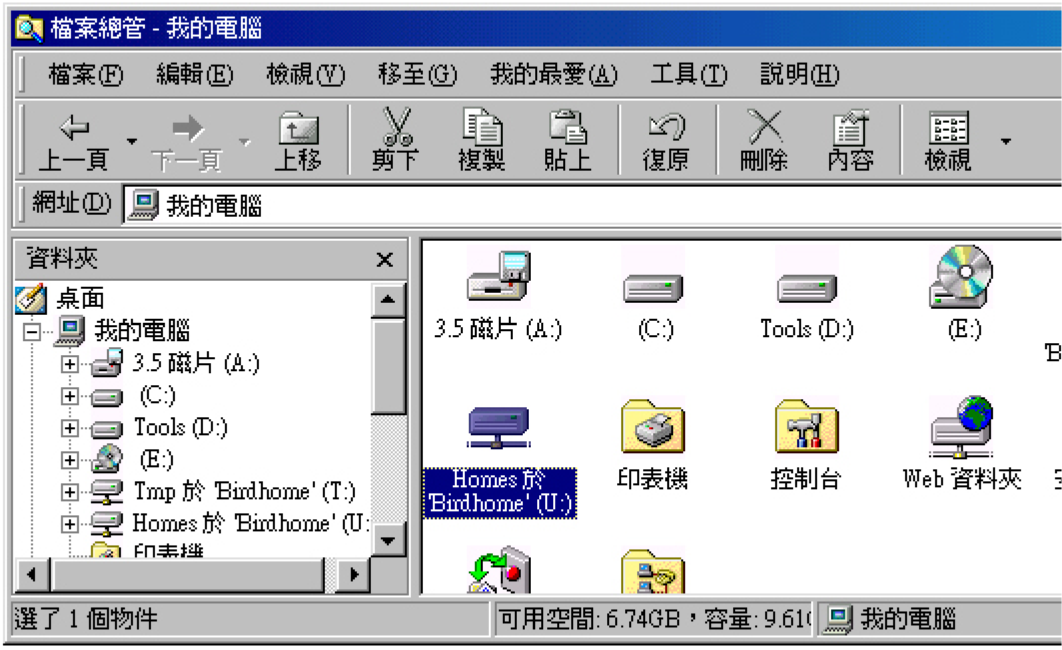
- 在上面出现的画面当中，只要动两个部分，分别是(1)勾选『登入 Windows NT 网域』这一项(因为 SAMBA 的 PDC 是仿真 Windows NT 的系统啊！)，并且在网域的地方填入您的网域，我们这里是 birdhouse 喔！然后按下确定，会回到前面的画面，而在前面的画面当中，选择『识别数据』会出现如下画面：



- 在上面的画面中，请务必填好计算机名称与工作群组呐！那个工作群组就是我们的网域 (birdhouse) 啊！然后按下确定，并且重新开机，重新开机后，会出现如下的登入画面喔！



只要填入正确的 PDC 上面的 ID 与密码，加上正确的网域，就可以登入了！而在启动 Windows 的画面时，会有一些程序在跑，那就是我们刚刚在 /home/samba/netlogon/startup.bat 所建立的批次档啰！跑完之后，您可以开启档案总管，就会发现如下画面喔：



没错！多了两个磁盘分割槽出来啰！恭喜您，又联机上啰！

所以说， Windows 98 要连上 PDC 很容易吧！几个动作就搞定了！

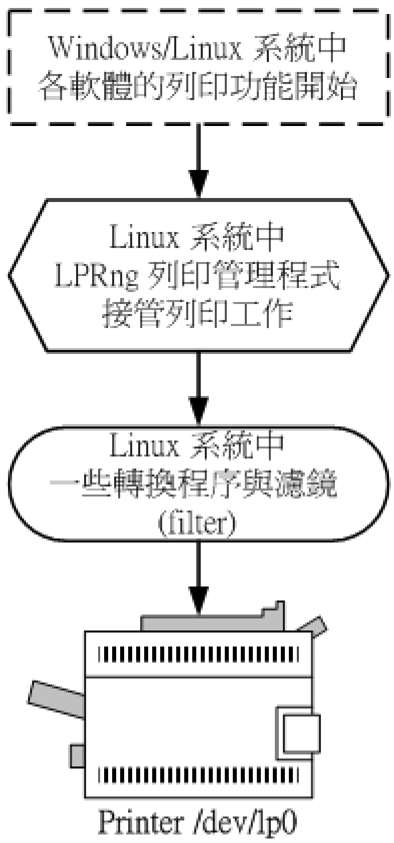
好了，关于 SAMBA 的 PDC 作法我们就谈到这里，还有更多的信息您可以前往这个章节最后面的『参考资源』所列出的网址去查阅，因为还有很多的作法呐！关于 PDC ，事实上，我觉得在一个网域当中，如果有多部的 Windows NT 主机，例如 Windows 2000 这一类的比较稳定的个人使用桌面版本时，使用 PDC 就很有用了！因为 Windows 2000 也是一个多人的操作系统，不像 Windows 98 是单人的操作系统，所以，当使用 Windows 2000 而无法登入 PDC 时，基本上，您是无法使

用 Windows 2000 上面的任何的信息的，但是在 Windows 98 上面若无法正确的登入，您仍然具有该计算机的主控权喔！所以，我个人是觉得 PDC 在 Windows 2000 主机比较多的环境下，用途会更广喔！如果只是 Windows 98 的环境，那么.....或许 PDC 还是不要玩吧！我觉得对于新手来说，好难呐！ @_@

设定成为打印机服务器 (printer server)

刚刚上面的说明大部分都是针对磁盘与目录的分享来进行 SAMBA 的设定，那么 SAMBA 有没有可能分享出装置呢？举个例子来说，例如那个我们在办公室都会用到的打印机 (Printer) 呢？如果说，能够让我们办公室的 File Server 同时提供打印机打印的功能，如此一来，藉由 Linux 这个操作系统可以 24 小时开机的稳定功能，我们就可以随时进行打印啰！没错！这真是个重要的任务啊！所以，底下我们就以 SAMBA 进行打印机打印功能的提供者的角度，来介绍这个 smb.conf 的设定喔！

事实上，在 Linux 底下的打印工作，是统一交由标准打印接口(或者说是程序)来进行数据与打印机之间的传输的，而这个打印的接口目前主要有两个，一个是比较老牌的 LPRng ，另一个则是功能较为强大的 CUPS (Common Unix Printing System, CUPS) 接口。因为 LPRng 的设定比较简单，而且功能也不差，所以这里我们主要是以 LPRng 这个打印管理程序来进行说明的喔！底下就是 LPRng 这个打印管理程序的一般工作流程：



整个打印流程是这样的：(1)当我们在 Windows 底下进行打印时 (就是按下打印按钮后)， (2)经由网络传输功能将打印的工作传至 Linux 系统下的 LPRng 这个打印程序来接管， (3)之后经过一些转换程序(转成正确的打印格式)以及滤镜 (filter) 功能，将数据转换成可以经由打印机输出的格式后， (4)最后就可以由打印机 (/dev/lp0) 来印出了！请注意喔，在 Linux 底下，打印机的装置代号为 /dev/lp0 ，第二部打印机则是 /dev/lp1 以此类推！

经过上面的程序说明后，我们知道要以 SAMBA 进行打印机的分享工作，实在很简单！您可以这样一步一步的进行喔！

40. 确定打印机可以正确输出：
    既然要分享打印机，当然就需要在 Linux 上面的打印机可以正确的打印咯！请这样做：
    - 将打印机接在 Linux 主机的 LPT 打印端口上 (就是 25 针的那个插槽)；
    - 打开打印机的电源 (以后就不要关了吧！ ^_^)；
    - 在 Linux 主机上面开一个终端机 (Terminal) ，然后输入：

    ```
    [root@test root]# echo "Hello world" > /dev/lp0
    ```

    - 照道理来说，这样就可以将『 Hello world 』这个字符串输出到打印机上面去了！
41. 如果在打印机上面印出了 Hello world 时，就表示打印机已经准备妥当了！接下来就是各种设定啰！

42. 安装 LPRng 套件：
    LPRng 是 Red Hat 主要的套件之一，所以如果您是 Red Hat 这个 Linux distribution 的使用者，请拿出您的原版光盘，然后就可以使用 RPM 来安装 LPRng 了！不过，如果您不是使用 Red Hat 的话，那么很可能因为您的 distribution 仅提供 CPUS 而已，所以就必须要手动的安装 LPRng 啰！安装的方法可以这样做：
    - 先到 LPRng 的官方网站下载最新的 LPRng 套件：
      官方网站：http://www.lprng.com
      FTP 网站：ftp://ftp.lprng.com/pub/LPRng/LPRng/
      在我这个例子当中，主要是以 LPRng-3.8.21.tgz 这个套件为准喔！
    - 将捉下来的套件到 /usr/local 底下解压缩，并且开始设定与安装！(因为我没有使用 SSL 加密以及 Kerberos 等功能，所以直接将他拿掉了！详细的功能参数请自行读取该目录下的 INSTALL ！)

    ```
    [root@test root]# cd /usr/local/src
    [root@test src]# tar -zxvf /root/LPRng-3.8.21.tgz
    .....(会产生一个名为 LPTng-3.8.21 的档案)...
    [root@test src]# cd LPRng-3.8.21

    [root@test LPRng-3.8.21]# ./configure --prefix=/usr/local/LPRng \
    > --disable-keros_checks --disable-ssl --enable-nls
    ```

```
....( 花一段时间来进行系统确认！ )....
建议可以使用 ./configure --help 来察看每一个设定项目的说明！
[root@test LPRng-3.8.21]# make clean all
....再花一段时间来等待～
[root@test LPRng-3.8.21]# make install   # 开始给他安装下去！
这个时候， LPRng 套件的所有数据都放置在/usr/local/LPRng 里面了！

[root@test LPRng-3.8.21]# vi /etc/man.config
# 在这个档案当中加入一行
MANPATH /usr/local/LPRng/man    # VBird 2003/07/20
# 这样未来在进行资料的 man 时，就可以直接查到 LPRng 的相关参数了！
# 例如 man checkpc、 man lprm ...

[roo@test LPRng-3.8.21]#vi /etc/profile
# 在这个档案当中， export PATH 该行之前，加入：
PATH="$PATH":/usr/local/LPRng/bin:/usr/local/LPRng/sbin
# 这样一来，未来在登入主机之后，就可以进行 LPRng 的相关指令功能！
```

▪

43. 设定 LPRng 的设定档：

    安装好了 LPRng 之后，接下来就是要使用 LPRng 的设定档来设定好您的打印机了！由于使用 RPM 安装与 Tarball 安装时，一些档案放置的地点不一样，所以，底下的设定在不同的系统当中可能会有点小差异喔！请依照您的系统作一下设定更改吧！（ 我这里分别以 Red Hat 9 这个使用 RPM 安装与 Mandrake 9.1 以 Tarball 安装时的设定为主 ）

```
0. 先确认 lp 这个系统账号是否存在?
[root@test root]# id lp
uid=4(lp) gid=7(lp) groups=7(lp)
万一出现了『 id: lp: No such user 』时，
请务必进行底下的动作，以新增打印机管理员账号：
groupadd lp
useradd -s /sbin/nologin -d /var/spool/lpd -g lp -r lp
mkdir -p /var/spool/lpd/lp0
chown lp:lp -R /var/spool/lpd
如果您的系统当中早就已经有了 lp 这个账号，那么上面的动作就不需要进行！

1. 先设定好 printcap 这个主要设定档！
[root@test root]# cp /etc/printcap /etc/printcap.bak.raw#只是备份
[root@test root]# vi /etc/printcap
# 如果是使用 Tarball 安装的，这个档案在 /usr/local/LPRng/etc/printcap
lp0|HP-LaserJet-1100:\
        :sh:\
        :ml=0:\
```

```
        :mx=0:\
        :sd=/var/spool/lpd/lp0:\
        :lp=/dev/lp0:\
#上面说明的是：
# lp0    为打印机在 Linux 的名称，HP-LaserJet-1100 为打印机 lp0 的别名
#        请注意， HP-LasetJet-1100 可以随意设定，但是 lp0 最好保留，
#        因为未来比较容易处理一些突发状况！
# sh     为是否印出标头，这里我们没有设定标头喔！
# ml, mx 为打印机能够接受的讯息，这里我们不加限制！
# sd     为打印工作的队列(打印档案暂存的目录)
# lp     就是打印机的实际装置代号了！
# 事实上，重点仅在于 lp 与 sd 而已～ ^_^

[root@test root]# checkpc -f
# 这个指令主要就是用来确认 printcap 是否设定正确的一个指令！
# 如果使用 Tarball 安装的朋友，请以 /usr/local/LPRng/sbin/checkpc -f 确认喔

2. 设定 LPRng 的主要存取设定档 lpd.perms
[root@test root]# vi /etc/lpd.perms
# 这个档案的内容挺重要的！因为涉及了其它使用者是否能使用您的 Linux 打印机
# 那个 perms 就是 permission 的意思啦！也就是说，谁能使用 lpd 这个 daemon
# lpd 就是管理打印机的 LPRng 的 daemon 啰！
# 1. 先以 IP 限制能来上来使用 Linux 的 Printer 的 IP 群，我这里开放出
#    192.168.0.0/255.255.255.0 这个网段喔！
REJECT SERVICE=X NOT REMOTEIP=127.0.0.1,192.168.0.0/255.255.255.0
#    如果您想要加入其它的 IP ，可以在后面以逗号继续分隔增加喔！
# 2. 再来是一些控制选项
ACCEPT SERVICE=C LPC=lpd,status,printcap,hold,release,stop,start REMOTEGROUP=bird
ACCEPT SERVICE=M SAMEHOST SAMEUSER
ACCEPT SERVICE=M SERVER REMOTEUSER=root
DEFAULT ACCEPT
#    事实上，由于我们在后头还会继续的设定防火墙，所以这里约略设定一下即可！

3. 设定一下启动的档案
[root@test root]# vi /etc/rc.d/init.d/lpd
# 如果是以 RPM 安装的话，这个步骤就不用作了！
# 使用 Tarball 安装时，预设没有这个档案，您可以写成这样：
#!/bin/sh
PATH=/usr/bin:/usr/sbin:/bin:/sbin:/usr/local/LPRng/sbin:/usr/local/LPRng/bin
export PATH
case "$1" in
  start)
        # Start daemons.
          echo "Starting lpd: \c"; /usr/local/LPRng/sbin/lpd;
```

```
        echo "";
        ;;
  stop)
      # Stop daemons.
        printf "Shutting down lpd: "
        pkill lpd
        echo "done"
        ;;
  *)
        echo "Usage: lpd {start|stop}"
        ;;
esac
```

44.
45. 启动打印机的服务：
    启动打印机实在很简单：

```
[root@test root]# /etc/rc.d/init.d/lpd start
[root@test root]# netstat -tln
tcp        0      0 0.0.0.0:515             0.0.0.0:*               LISTEN
[root@test root]# echo "/etc/rc.d/init.d/lpd start" >> /etc/rc.d/rc.local
```

46. 如果有出现 printer 或者是 515 这个 port number 的话，那就表示 lpd 应该已经在 LISTEN 了！如果确定该程序没有问题，就可以将他写入开机时启动的设定档当中啰！

47. 编写 smb.conf ，加入打印机的支持：
    既然打印机已经 ready 了，接下来就是要重新的给他设定好 smb.conf 啰！您可以这样加入一段支持：

```
[root@test root]# vi /etc/samba/smb.conf
# 不管在哪里，找到 smb.conf 然后编辑他就是了！然后加入底下这一段：
[global]
   printcap name = /etc/printcap
   load printers = yes
   printing = lprng
# 上面这三行预设应该会存在 smb.conf 当中，只要找到这三行，
# 并将行首的 ; 拿掉即可！注意，这三行是新增的！

# 特别给他留意一下，打印机的分享是经由 [printers] 相关的设定来提供的！
[printers]
   comment      = HP LaserJet 1100
   printable    = yes
   browsable    = no
```

```
   public       = no
   validusers   = bird puma addida amani pada
   printing     = lprng
   path         = /var/spool/lpd/samba

# 注意一下，上面重要的地方在于：
# printable  需要启动为 yes
# validusers 有需要的话就请自行设定，我这里仅允许五位使用者登入而已！
# printing  设定成使用 lprng 这个管理程序
# path      这个就是打印机队列，我将他设定在 /var/spool/lpd/samba 中

[root@test root]# mkdir -p /var/spool/lpd/samba
[root@test root]# chown root:root /var/spool/lpd/samba
[root@test root]# chmod 777 /var/spool/lpd/samba
[root@test root]# testparm
[root@test root]# /etc/rc.d/init.d/smb restart
```

48. 如此一来，在同一个网域的朋友就可以看到我这一部 Linux 分享的打印机了，并且，打印机的名称为 HP LaserJet 1100 喔！很方便吧！

49. 一些问题克服 (lprm, lpq...)
由上面的步骤进行来看，这个时候您的 Linux 主机应该已经可以分享您的打印机了！很开心吧！ ^_^ ！不过，我们还是得要稍微熟悉一下在 Linux 底下管理打印机的手段喔！为什么呢？想象一个画面，今天，您已经在 Windows 上面按下『打印』按钮，偏偏竟然是印错了档案，这个档案还有 100 页之多～因为 SAMBA 会先将打印数据队列在 /var/spool/lpd/samba 当中，所以，这个时候使用 Windows 并不能管理该打印机～哇！难道我要痴痴的等待 100 页印完吗？！呵呵！当然不需要！您可以登入 Linux 来控管您的打印数据呢！前提之下是，您必须使用与 SAMBA 相同的使用者登入系统喔！

```
[root@test root]# lpq
# 可以察看打印机的状态
Printer: lp0@test 'HP-LaserJet-1100'
 Queue: no printable jobs in queue
 Status: job 'cfA209test.vbird.idv.tw' removed at 13:29:24.163
[root@test root]# lprm all
# 将所有打印机的等待打印的数据通通删除！
```

50. 基本上，我最常下达的指令就是 lprm all 了！因为打印机出现问题时，使用这个指令可以将打印机等待中的数据通通杀掉，然后重新启动打印机，就可以恢复正常的打印啰！

所以将您的 SAMBA 设定成为打印机服务器也不难吧！ ^_^

设定成为打印机服务器 (Printer Server + CUPS 系统)

除了 LPRng 这个打印程序之外，我们还可以使用较新的 CUPS 系统 (Common Unix Printer System) 来做为我们的打印程序喔！底下我以 Red Hat 9 为例来设计 CUPS + SAMBA 的设定方式：

51. 安装 CPUS 的相关套件：
    安装的套件就是 CUPS 咯！检查一下吧！

```
[root@test root]# rpm -qa | grep cups
cups-libs-1.1.17-13.3
cups-1.1.17-13.3
qtcups-2.0-15
```

52. 在这个 cups 的套件当中，最重要的就是在 /etc/cups 这个目录内的档案了！尤其是主要的设定档 /etc/cups/cupsd.conf 喔！

53. 设定 CUPS 的设定档 cupsd.conf 并启动 cups：
    由于 CUPS 是一个服务，他可以对 Internet 提供打印的服务喔！所以，未来启动 cups 之后，会产生一个监听的接口，而谁可以登入这个监听的接口呢！？呵呵！没错，就是需要在这个 cupsd.conf 档案内设定啦！我们假设我们的 CUPS 是对内部的 192.168.0.0/24 这个网域开放的，所以：

```
[root@test root]# vi /etc/cups/cupsd.conf
# 其实这个档案的设定很类似 httpd.conf 呢！我们只要针对两个参数来设定即可，
# 分别是 / 与 /admin 喔！设定只有 192.168.0.0/24 可以登入！
<Location /admin>
AuthType Basic
AuthClass User
Order Deny,Allow
Deny From All
Allow From 192.168.0.0/24
# 注意一下，上面的 AuthClass User 这个设定值，可以让您以 Linux 上面的
# User 身份来登入 CUPS 以设定 Printer 喔！
# 这个 /admin 的设定内容主要与『打印机管理员』有关的设定
</Location>

<Location />
Order Deny,Allow
Deny From All
Allow From 192.168.0.0/24
```

```
# 这个 / 设定值则是决定谁可以登入使用 Printer 呢！
</Location>

[root@test root]# /etc/rc.d/init.d/cups start
[root@test root]# netstat -utln | grep 631
tcp        0      0 0.0.0.0:631             0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:631             0.0.0.0:*
# 请注意到，启动了 CUPS 之后，会产生这个 631 的埠口呐！
```

54.
55. 设定您的第一部打印机：
    好了，既然 CUPS 这样就可以设定成功了，接下来当然就是需要设定我们的打印机了！设定打印机可以很简单的以 lpadmin 来设定，也可以使用 Web 接口来设定喔！在文字接口下，假设您的打印机是 HP 的激光打印机 (lasetjet)，例如鸟哥研究室的 hp 1100 laserjet ，我可以这样设定：

```
[root@test root]# lpadmin -p HP-1100-LaserJet -m laserjet.ppd -E \
> -v parallel:/dev/lp0
# 参数说明：
# -p 后面接打印机名称，名称可以随便取！
# -m 后面接打印机的接口模块档案(module interface file)
# -E 则是说，这个打印机可以用来进行打印之意！
# -v 后面接的则是『串行端口 parallel』或者是『网络 socket 』！

# 以 lpadmin 设定好打印机后，整个打印机的状态列其实被放置在
# /etc/cups/printers.conf 里面喔！
```

56. 除了这个方法之外，其实我们可以使用简单的 Web 界面来管理，例如我可以在 192.168.0.100 这个 Client 上面，连上刚刚设定好打印机的 192.168.0.5 这个 server，连接的方法为使用浏览器，在网址列输入 http://192.168.0.5:631 ，然后按下 Printers 之后，出现下列图示：



注意一下，如果如同上面的图示中，出现了『file:/dev/null』呵呵！那就表示『您的打印机发生错误了』，因为一般来说，Linux 本机打印机应该是会出现：『parallel:/dev/lp0 』才对喔！在上面的图示当中，您可以选择『Modify Printer』或者是『Configure Printer』来最佳化您的打印机呐！ ^_^

57. 编写 smb.conf ，加入打印机的支持：
既然打印机已经 ready 了，接下来就是要重新的给他设定好 smb.conf 啰！您可以这样加入一段支持：

```
[root@test root]# vi /etc/samba/smb.conf
# 不管在哪里，找到 smb.conf 然后编辑他就是了！然后加入底下这一段：
[global]
   printcap name = cups
   load printers = yes
   printing = cups

[printers]
   comment       = HP LaserJet 1100
   printable     = yes
   browsable     = no
   public        = no
   validusers    = bird puma addida amani pada
   printing      = cups
   path          = /var/spool/lpd/samba
```

```
[root@test root]# mkdir -p /var/spool/lpd/samba
[root@test root]# chown root:root /var/spool/lpd/samba
[root@test root]# chmod 777 /var/spool/lpd/samba
[root@test root]# testparm
[root@test root]# /etc/rc.d/init.d/smb restart
```

58. 如此一来，在同一个网域的朋友就可以看到我这一部 Linux 分享的打印机了，并且，打印机的名称为 HP LaserJet 1100 喔！很方便吧！

在打印机的设定当中，鸟哥曾经发生过一件相当糗的事情，我拼了老命的设定 LPRng 以及 CUPS 就是无法设定好 Printer ，虽然已经连上打印机了，但是就是印出来都是乱码，很伤脑筋～等到我花了一整天去恶搞之后，最后竟然发现，错误的地方在于『BIOS 的并行端口设定』唉！伤脑筋的很～还记得每次开机的时候系统都会去读取 BIOS 的设定吗？一般来说，按下 DEL 按键后，会进入 BIOS 的设定画面，在该画面当中，选择相关的设定参数，与 Parallel 有关的项目，将『模块』改成 EPP/SPP 吧！这样就可以支持您的打印机了！天呐！就因为这个设定值，让鸟哥花了一两天的时间，还差一点将主机砸掉....

Client 端的设定：

整个 SABMA 的主要目的其实是针对局域网络来达成更便利的数据传输的手段，而既然是针对局域网络（ LAN ）的话，那么整个局域网络的 Windows 与 Linux 计算机的设定就得好好的搞定一番啦！因为整个 LAN 里面使用最多的大概就是那个 NetBIOS （ 一般常见的是后来的升级版，也就是 NetBEUI ），所以，您必须要在 Windows 里面至少设定两个通讯协议才行：

- TCP/IP
- NetBEUI

关于 Windows 的通讯协议设定方法，我们已经在前面的『局域网络简介』里面提过了，这里不再赘述，请自行前往参考。除了这个协议的设定之外，在同一个网域之内，要将计算机以 LAN 兜在一起，最好还需要网络识别的方式：

- 所有的计算机都是同一个『工作群组，Workgroup 』
- 所有的计算机都有独特的『NetBIOS 主机名称』。

还是得再次的强调，NetBIOS 主机名称与 Internet 上面使用 DNS 解析的主机名称不同！假设这个设定的项目通过了，那么该如何在 Windows 与 Linux 之间分享档案呢？

在 Windows 上浏览 Linux 分享档案的设定

在 Windows 之间分享档案真的是很简单，只要设定好『资源共享』或者是 Windows NT 系列的所谓的『共享』那么就可以使用彼此的档案啰！而要在 Windows 上面浏览 Linux 的 SAMBA 主机，那也很简单，只要知道 Linux 的 NetBIOS Name 之后，直接在『网络上的芳邻』去点选，也可以利用『开始』=>『搜寻』=>『计算机』=>填写入 Linux 的 NetBIOS Name ，如果一切联机都没有问题，就可以连接上 SAMBA 主机啦！因为实在太简单了！所以这里我们就不再强调了！(注：事实上，使用档案总管就可以进行很多分享的工作了！尤其是『联机网络磁盘驱动器』的项目喔！)

在 Linux 上浏览 Windows 分享档案的设定

既然 Windows 可以浏览 Linux 的档案，没道理 Linux 不能浏览 Windows 的档案吧！呵呵！这么说真是不错啊！这里我们介绍几个 Samba 提供的指令，来让您轻松的使用 Windows 的资源喔！

```
smbclient 语法介绍：

1. 察看 NetBIOS 主机分享的目录状态：
[root@test root]# smbclient -L \\netbiosname [-U username]
# 这个 -L 的参数主要的目的在于察看 netbiosname 那部主机有提供什么目录，
# 比较需要留意的是，如果没有加上 -U username 时，那么预设是以匿名者
# anonymous 来登入该 NetBIOSname 的，所以能看到的目录或许就会被限制喔！
# 另外，NetBIOS 主机名称在 Linux 底下，需要在前面加上『 \\ 』喔！
范例：
[root@test root]# smbclient -L \\bird2000 -U bird
added interface ip=192.168.1.2 bcast=192.168.1.255 nmask=255.255.255.0
Password:
Domain=[BIRDHOUSE] OS=[Windows 5.0] Server=[Windows 2000 LAN Manager]

        Sharename      Type      Comment
        ---------      ----      -------
        IPC$           IPC       远程 IPC
        D$             Disk      预设共享
        ADMIN$         Disk      远程管理
        C$             Disk      预设共享

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
# 如上面的范例来看，不管那个 \\bird2000 是 Linux Samba 还是 Windows ，
# 都可以显示出类似上面的画面，在 sharename 的地方，显示 bird 这个使用者
# 登入时，可以取得使用的目录！
```

```
2. 信差服务：
[root@test root]# smbclient -M netbiosname
# 这个 -M 是 messages 的意思，他可以将数据以 Windows 的 winpopup 程序
# 来传给该计算机上面的使用者喔！除非对方的 winpopup 关闭！
范例：
[root@test root]# smbclient -M bird2000
added interface ip=192.168.1.2 bcast=192.168.1.255 nmask=255.255.255.0
Connected. Type your message, ending it with a Control-D
Hello, How are you?  <==最后这里是按下 Ctrl + D 来结束的！
sent 21 bytes
# 然后 bird2000 这部计算机会出现如下画面：
```




```
3. 登入目录：
[root@test root]# smbclient '\\netbiosname\directory' [-U useranem] [-W workgroup]
# 这才是 smbclient 最主要的目的啦！就是登入 NetBIOS 那部主机的某个目录，
# 然后可以使用 ftp 的功能来将数据上传或下载喔！
范例：
[root@test root]# smbclient -L \\127.0.0.1 -U bird
added interface ip=192.168.1.2 bcast=192.168.1.255 nmask=255.255.255.0
Password:
Domain=[BIRDHOUSE] OS=[Unix] Server=[Samba 2.2.7a-security-rollup-fix]

        Sharename      Type      Comment
        ---------      ----      -------
        netlogon       Disk      Network Logon Service
        public         Disk      Public Stuff
        IPC$           IPC       IPC Service (Bird's testing SAMBA Server)
        ADMIN$         Disk      IPC Service (Bird's testing SAMBA Server)

        Server               Comment
        ---------            -------
        BIRDHOME             Bird's testing SAMBA Server

        Workgroup            Master
        ---------            -------
        BIRDHOUSE            BIRDHOME
```

```
# 先以 -L 的参数察看一下 127.0.0.1 这个 IP 的分享情况，发现 bird 可以登入
# \\127.0.0.1\public 这个目录喔！由于 \ 在 bash 里面是有特殊意义的字符，
# 所以我们要以单引号 ' 将目录整个括号起来喔！
[root@test root]# smbclient '\\127.0.0.1\public' -U bird
added interface ip=192.168.1.2 bcast=192.168.1.255 nmask=255.255.255.0
Password:
Domain=[BIRDHOUSE] OS=[Unix] Server=[Samba 2.2.7a-security-rollup-fix]
smb: \> dir
# 在 smb 底下其实就是在 \\127.0.0.1\public 这个目录底下啦！所以，
# 我们可以使用 dir，get，put 等常用的 ftp 指令来进行数据传输了！
?   :列出所有可以用的指令，常用！
cd  :变换到远程主机的目录
del :杀掉某个档案
lcd :变换本机端的目录
ls  :察看目前所在目录的档案
dir :与 ls 相同
get :下载单一档案
mget:下载大量档案
mput:上传大量档案
put :上传单一档案
rm  :删除档案
# 其它的指令用法请参考 man smbclient 喔！
```

事实上，使用 smbclient 一点也不方便，因为使用的是 ftp 的功能语法，有点怪怪的～能不能像 Windows 那样，可以直接联机网络磁盘驱动器啊？！这当然没有问题！不过就需要藉由 smbmount 来协助了！smbmount 可以将远程主机分享出来的目录整个给他 mount 到本机的 mount point(某个目录)，如此一来，远程主机的目录就好像在我们本机的一个 partition 一样喔！可以直接执行复制、编辑等动作！这可就好用的多了！底下我们来谈一谈怎么用这个 smbmount 吧！

```
smbmount 语法介绍：

[root@test root]# smbmount \\netbiosname\directory [-o options]
参数说明：
netbiosname ：可以是 IP 也可以是网芳上面的 NetBIOS 主机名称
-o 后面接的参数 options 常用的有底下这些参数：
  username=你的登入账号：例如 username=bird
  password=你的登入密码：需要与上面 username 相对应啊！
  codepage=语言格式：这个可以设定支持的语系，例如繁体中文：codepage=cp950
范例：
# 假设我要以 bird 身份，密码为 mypasswd 挂载远程主机 \\birdhome\tmp
# 那个目录，并且挂载到我 Linux 的 /home/birdhome 这个目录，如何做？
[root@test root]# smbclient -L \\birdhome -U bird
# 先以 smbclient 找出可以挂载的目录！在这个案例中，我有 \\birdhome\tmp
```

```
# 可以挂载！
[root@test root]# smbmount '\\birdhome\tmp' /home/birdhome \
> -o username='bird',password='mypasswd',codepage='cp950'
# 再次给他强调一下，因为 \ 在 bash 当中是特殊字符，所以挂载时请特别
# 使用 ' 来将 \ 设定成为一般字符！还有，在 -o 后面的各项参数中，
# 中间都是以逗号来隔开的！并且设定值最好也使用单引号 ' 来设定！
[root@test root]# df
//birdhome/tmp         3020160    186880   2833280   7% /home/birdhome
# 如上所示，你应该就可以看到目录对应 mount point 啦！
```

经由 smbmount 的动作，我们就可以轻易的将远程分享出来的咚咚给他挂载到自己 Linux 本机上面！好用的很～事实上，原本 mount 这个指令如果有支持 smbfs 的话，那也就可以直接挂载 网络上的芳邻 分享出来的目录了！例如上面的例子当中，我们也可以这样下达指令：

```
[root@test root]# mount -t smbfs '\\birdhome\tmp' /home/birdhome   \
> -o username=bird,password='mypasswd',iocharset='cp950'
# 上面那个 iocharset 与 codepage 都是用来设定语系的！
```

更详细的 mount 用法，请 man mount 或者看考『鸟哥的 Linux 私房菜 -- 基础学习篇』的内容喔！ ^_^

安全相关方面

使用 SAMBA 其实也是挺有一定的危险性的！因为近期来利用 NetBIOS 来进行攻击的病虫实在是不少，而且，也有很多坊间的书籍在告诉大家『入侵的手段』啊！所以，我常常告诫我的朋友们，在 Windows 上面，资源共享完毕之后，应该立即将该分享的权限取消！以避免不小心被入侵或者被感染病毒的机会啊！此外，如果您是学校单位的话，那么利用 SAMBA 分配给每的班级自己的网页空间，应该是不错的想法，不过，如果有某个班级一下子就用掉您的大部分硬盘空间，这样对其他的使用者来说，有点不太公平～因此，利用 quota 来限制每个使用者(班级)的硬盘使用空间，也是一个很可行的方案啊！ ^_^

配合 quota 来规范使用者的使用空间

quota 是磁盘配额限制的一个 daemon ，您可以依据不同的使用者来加以限制他们能够使用的硬盘空间，前提是『该磁盘空间必须是一个独立的 partition 』才行，不建议针对根目录『 / 』进行 quota ！这也是为什么我们在主机规划时特别建议大家独立出一个 partition 来进行硬盘规划的原因！关于 quota 的详细用法在『鸟哥的 Linux 私房菜--基础学习篇』里面已经谈得很清楚了，这里我们不再详谈！只给大家来做个练习就是了！

习题练习

问：在规划主机的时候，我将 /dev/hda2 独立一个 partition 给 /home 这个目录，现在想要规划 quota 针对 bird 这个使用者进行磁盘配额的限制，他的 hardlimit 为 50mb 而 softlimit 为 40mb ，请问整个动作应该如何进行？

1. 编辑修改 /etc/fstab ，使得 /dev/hda2 这个 partition 成为如下模样：
/dev/hda2 /home ext3 defaults,usrquota,grpquota 1 1
修改完毕之后，请千万记得使用 mount -a 查询看看有没有设定错误（上面总共只有六个字段喔！)，这个动作如果发生错误，那么请记得再次的进行 /etc/fstab 的修改！请特别留意啊！因为写错的话，很容易造成无法正常开机的窘境！

2. 上述动作确定没有问题之后，请重新开机启动 quota 的支持：
sync; sync; sync; reboot

3. 执行底下的指令，建立 quota 所需要的设定档案：
quotacheck -avug
特别注意，如果上述的动作发生『找不到设定文件』的意思的讯息时，您可以这样做：
touch /home/aquota.user; touch /home/aquota.group
然后在进行一次 quotacheck 即可！

4. 启动 quota 啰：
quotaon -av

5. 设定 bird 的磁盘配额：
edquota -u bird
```
Disk quotas for user test (uid 501):
  Filesystem   blocks     soft  hard inodes  soft  hard
  /dev/hdb2        32   40000  50000      8     0     0
```
因为我们只针对硬盘空间，不针对 inode 做限制，所以只要前面的数字修订即可！请注意，数字的单位为 kbytes ！

如何设定防火墙 iptables

如果您的主机环境本身已经具有防火墙了，而且是较为严密的防火墙，那么 SAMBA 所需要的 port 应该会被您挡住才对～这个时候，请特别在您的防火墙规则中开放出 SAMBA 所需要的 port 与针对的 IP 网域喔！一般来说，因为 SAMBA 越来越不安全了(前面说过，病毒的问题很严重)，所以，通常我们会建议大家 SAMBA 仅针对我们自己的网域来开放即可！甚至仅针对少部分主机来开放即可呢！

要设定防火墙，请务必具有 iptables 的基本观念，在 认识网络安全 的章节当中我们已经谈过了，请自行前往参考喔！在这里，我们仅针对 SAMBA 所需要的 port 与 IP 网段进行说明。

假设：

- 仅针对 192.168.0.0/24 这个网域开放 SAMBA；

- SAMBA 的 port 有 137~139 tcp/udp;
- SAMBA 主机的网络卡为 eth0

针对上面的设定，您可以在防火墙机制当中加入两行喔：

```
/sbin/iptables -A INPUT -p tcp -i eth0 -s 192.168.0.0/24 --dport 137:139  -j
ACCEPT
/sbin/iptables -A INPUT -p udp -i eth0 -s 192.168.0.0/24 --dport 137:139  -j
ACCEPT
```

当然啦！这是很简单很简单的防火墙规则，您必须要依据您的环境自行修改(通常修改那个 192.168.0.0/24 网段即可！)。除了 iptables 必须要开放之外，您还需要将管制 daemon 是否能够进出的 TCP Wrappers 开放才行呐！如果万一您看到 /etc/hosts.deny 里面多了这一行：

```
ALL : ALL
```

不要怀疑，很可能会造成不只 SAMBA 不能动作而已～连其它的 daemon 都可能会无法动作呢！所以，请将这一行拿掉吧！或者是将适合的 daemon 开放在 /etc/hosts.allow 档案里面喔！

如何设定 daemons 的抵挡功能 (hosts allow 项目)

除了 iptables 与 TCP Wrappers 这两个 Linux 本身的防火墙机制之外，其实 SAMBA 本身也就提供了防火墙的机制啦！那就是在 /etc/samba/smb.conf 这个档案里面的 hosts allow 与 hosts deny 的项目了!这两个设定是在 [global] 里面的设定，一般来说，只要设定 hosts allow 那一项就可以了!因为没有被设定到里面的 IP 都无法使用 SAMBA 喔!以上面的防火墙例子来说，我们仅开放 192.168.0.0/24 这个网域可以使用 137~139 的 port ，不过，在这个网域当中，我们又仅针对五部计算机开放服务，例如 192.168.0.1~192.168.0.5 ，那么我可以在 smb.conf 里面多加这一行：

```
hosts allow = 192.168.0.1 192.168.0.2 192.168.0.3 192.168.0.4 192.168.0.5
```

各个 IP 之间都以空格隔开即可！此外，如果您想要针对整个网域来开放，例如本机的 127.0.0.0/8 这个网域，那么可以再加入：

```
hosts allow = 192.168.0.1 192.168.0.2 192.168.0.3 192.168.0.4 192.168.0.5 127.
```

这样就可以了！如此一来，虽然本机所在的网段 192.168.0.0/24 都可以尝试登入 SAMBA (因为可以通过 iptables 的规则)，不过却会被 SAMBA 本身的设定挡住！这有什么好处啊？！好处就是在网段里面尝试登入 SAMBA 但是却遭 SAMBA 拒绝的 IP 与原因，都会被纪录到登录文件里面去，如此一来，我们可以被动的利用分析 SAMBA 的登录档，以发现是否有人想要入侵或者是有某部计算机是否已经被不正常的病毒所感染呢！

要备份些什么咚咚？(/home, 开放的咚咚, smb.conf, smbpasswd...)

跟 SAMBA 最有关的当然就是使用者的信息啰！所以，您应该要备份的就是使用者的数据，以及 Samba 相关的设定数据了！比较相关的有几个咚咚：

- /etc (因为含有 /etc/passwd 以及 /etc/samba 里面的资料，所以建议可以全部给他备份下来)
- /home (还是可以察看一下，是否有其它非必要的！不过，一般来说，都会是备份的个人数据比较多啦！)
- 其它由 smb.conf 里面设定开放的目录。

无论如何，备份还是挺重要的！

问题克服

通常我们在设定 SAMBA 的时候，如果是以 Workgroup 的方式来进行 smb.conf 的设定时，几乎很容易就可以设定成功了！并没有什么很困难的步骤！不过，万一还是无法成功的设定起来，请务必察看登录档，也就是在 /var/log/samba 里面的数据！在这里面的资料当中，您会发现：咦！怎么这么多档案啊！因为我们在 smb.conf 里面设定了：

```
log file = /var/log/samba/%m.log
```

那个 %m 是指 Client 的 NetBIOS Name 的意思，所以，当有个 bird2000 的主机来登入我们的 birdhome 主机时，那么登入的信息就会被纪录在 /var/log/samba/bird2000.log 档案喔！而如果万一来源 IP 并没有 Netbios name 的时候，那么很可能是一些错误讯息，这些错误讯息就会被纪录到 log.smbd 里面去了！所以，如果您要察看某部计算机连上您的 SAMBA 主机发生了什么问题时，特别要留意这个登录档的形式喔！

另外，如果您的 SAMBA 明明已经启动完成了，却偏偏老是无法成功，又无法查出问题时，建议先关闭 Samba 一阵子，再重新启动：

```
/etc/rc.d/init.d/smb stop
```

在我的案例当中，确实有几次是因为 PID 与 NetBIOS 的问题，导致整个 SAMBA 怪怪的～所以完整的关闭之后，经过一阵子的短暂时间，再重新启动，应该就可以恢复正常了！

还有，万一您在进行写入的动作时，老是发现『您没有相关写入的权限！』，不要怀疑，几乎可以确定是 Permission 的问题，也就是 Linux 的权限与 SAMBA 开放的权限并不相符合！无论如何，您必须要了解能不能写入 Linux 磁盘，看的是 PID 的权限与 Linxu 档案系统是否吻合，而那个 smb.conf 里面设定的相关权限只是在 SAMBA 运作过程当中『预计』要给使用者的权限而已，并不能取代真正的 Linux 权限喔！所以，万一真的发现该问题存在，请登入 Linux 系统，查验一下该对应的目录的 permission 吧！ ^_^

附带说明一点，常常有朋友会问 swat (Samba Web Administration Tool)是什么？那个其实是

SAMBA 提供给 SAMBA 系统管理员的一个 Web 图形接口的管理工具！我个人是不太喜欢使用工具来工作的，如果您有兴趣，请自行参考相关的书籍来使用吧！ ^_^

重点回顾

- (等待更新中)

本章与 LPI 的关系

- 在 LPI 网站 http://www.lpi.org 里面提到的，(等待更新中)

参考资源：

- man 5 smb.conf
- Study Area ：http://www.study-area.org/linux/servers/linux_samba.htm
- 电子书 Using Samba：http://de.samba.org/samba/ftp/docs/htmldocs/using_samba/index.html
- Samba PDC FAQ：http://de.samba.org/samba/ftp/docs/htmldocs/samba-pdc-faq.html
- Samba PDC HOWTO: http://de.samba.org/samba/ftp/docs/htmldocs/samba-pdc-howto.html
- SAMBA 官方网站：http://www.samba.org/
- rondo 的 SAMBA 密技：http://rondo.study-area.org/~linux/student_samba/server/samba/
- 依玛猫的打印文件：http://www.imacat.idv.tw/tech/lnxprint.html

本章习题练习

- 一般来说， SAMBA 使用的设定档放置在哪里？档名为何？
- 哪一个指令可以用来判断 smb.conf 这个设定档的正确性？
- 哪一个指令可以用来察看 SAMBA 主机分享出什么目录？
- smbmount 的功能为何？
- 我今天使用 smbpasswd 去新增一位使用者 badbird，让他可以登入我的 Linux SAMBA 主机，但是无论如何就是无法新增。您认为原因可能是什么？

简易 Mail Server 架设 -- Sendmail 设定

最近更新日期：2003/09/12

在这个邮件服务器的架设中，我们首先谈论 Mail 与 DNS 的重要相关性，然后依序介绍 Mail Server 的相关名词，以及 Mail Server 的运作基本流程与协议，也会谈到相关的 Relay 与 邮件认证机制 等项目，这些项目对于未来邮件主机的管理与设定是重要的，请不要忽略了这方面问题的讨论喔。当然，主要的目的还是在于架设 Sendmail 这个使用最为广泛的邮件主机服务器软件啰！这里我们以 Red Hat 7.x 以及 Red Hat 9 为主体来说明 Sendmail 的主要架构，要说明的是，虽然本文是以 Red Hat 为主体，但是 Sendmail 的架构仍然可以在其它使用 Sendmail 的 Linux 系统当中成立的。而重头戏则在最后面的 Tarball 安装一套完整的 Sendmail 喔 (我是以 Mandrake 9.0 及 Red Hat 7.x 版本来测试的)！如果您的 Linux 上面本来就没有 Sendmail ，并且您还是习惯 Sendmail 这个套件，那么，这篇文章仍然相当的适合您查阅！

：2. 关于 quota 的设定与 /var/spool/mail 目录的转移

本章与 LPI 的关系

参考资源：

本章习题练习

前言：

电子邮件带来的好处：

在目前的社会当中，没有电子邮件 (e-mail) 似乎是蛮奇怪的一件事！。可以说，现在 e-mail 已经成为一个很普遍的人与人之间的沟通管道了，电子邮件可以很快速的帮你将文件或讯息传送到地球上的任何一个有网络存在的角落，当然，你也可以在任何有网络的地方，连上 Internet 去收取你的信件！很快乐不是吗？是的！e-mail 的存在是相当重要的，你可以藉由这个电子邮件取得最实时的一手数据！你也可以利用他帮你联络好朋友，还可以用来把马子哩！君不见前一阵子相当有名的『电子情书, You got a mail』这部电影吗？呵呵！反正，电子邮件真的带给目前繁忙的人们一个相当轻松获得信息的方式！

电子邮件衍生的问题：

不过，遗憾的是，只要是有人类的地方，就会有很多你意想不到的事情会出现了，当然，e-mail 也不例外，怎么说呢？ 我们来慢慢的分析一下吧：

1. 电子邮件夹带病毒：你可以常常听到电子邮件所夹带的病毒对吧！没错，利用电子邮件以及人们对于电子邮件的漫不经心的态度，使得以电子邮件为媒介的计算机病毒更容易『深入人群』当中呐！这个问题造成大大小小的伤害，如果发生在大企业当中，那可真是受不了那～那个主管受的了一天到晚计算机重新安装的～而且万一中毒的是大型主机，光是数据的损毁就可能让公司倒闭了....
2. 怪客入侵事件：没错！ e-mail 也是一个相当不安全的网络协议，你可以轻易的使用怪客软件 (Cracker) 就可以取得使用者在利用 e-mail 传送过程当中的，将他的账号与密码撷取下来，分析之后，并进一步的破解对方的邮件主机～哇！真是乱可怕一把的！
3. 广告与垃圾信件：这个可说是目前各大 ISP 心中永远的痛～这些垃圾信件可以占掉很多那少的可怜的频宽，使得正常使用者联机速度与质量下降，更可能造成网络的停顿～当然，常常收到垃圾信件的你，大概也不好过吧！
4. 暴力攻击事件：万一你没有将邮件主机设定好，嘿嘿！送信者可以藉由你主机收信的功能，发送大量的信件，让你『一次收个够！』灌爆你的主机硬盘，想要不当机都粉难～
5. 真实社会的讨厌事情：『黑函』！听到会不会很害怕？当然很害怕啦！偏偏，使用 e-mail 就可以作很多的坏事～这真是太不道德了～
6. 不实的信件内容：只要注意到消基会的讯息就可以知道啦，不明来源的电子邮件说的内容，嘿嘿！不要轻易的相信，因为，很多可是以讹传讹，结果，大家都被耍了的～例如，你的朋友收到一封信，认为『哇！这是大事情』，所以在没有求证的情况下，将信『转寄』给你看，嘿！你的朋友寄给你的，当然要相信他啦！立刻再转寄，如此一再地循环，嘿嘿！这个错误内容的讯息马上就让大家知道，更可怕的是『还会让大家接受～』所以，看到任何讯息时，请千万要记得求证一下呐！

可怕吧！电子邮件会衍生出这么多的问题说～

网管人员的痛：

因为 e-mail 的便利性，与邮件主机的容易受到恶意的攻击，这两个事件真的是很难分的清，怎么说呢？如果为了使用者的便利性而大开便利之门，那么，您的邮件主机大概用不了多久，就会被列入黑名单之中，进而成为大家的拒绝往来户，呵呵！反而成为笑柄，并且反而造成了使用者的不便～而如果管制得太严格，又显的不够人性化～最起码，你的主管就会不太满意～怎么办呢？！哈哈！没错啦！邮件主机就是这么回

事～让人又爱又怕的一个玩意儿，搞的定他，恭喜你，一切圆满顺利，搞不定他，主机被当成垃圾信件转运站事小，丢掉工作可就『兹事体大』啦～就因为他是这么重要，但是又这么难以搞定，所以啦，我们可要好好的学一学他呐！

邮件服务器运作原理：

既然要使用 e-mail ，当然就需要邮件主机服务器啰 (Mail Server)！不然你的信要怎样寄出去呢？事实上，mail server 的原理说难不难，但是说简单吗～似乎又有点难以理解ㄋㄟ～，所以，底下我们要来谈一谈他的原理部分，然后再针对主机的设定来进行说明咯！底下，我们首先要讲的，就是『Mail server 系统与 DNS 系统有什么关连性？』这个部分新手最容易被搞混哩，是否要架设 mail server 就『宿命』的一定得架设 DNS 主机在你的主机上面吗？

Mail 与 DNS 系统的相关性：

一直以来，Mail server 与 DNS 系统就是分不开的，怎么说呢？今天如果你要寄电子邮件的话，那么就得藉由邮件主机帮你将信件送出去，对吧！那么我们在 DNS 那个篇幅里面也谈到了相当多的概念了，就是，人脑实在无法记忆住计算机网络的 IP 数据，因此，才会有所谓的 Domain Name System, DNS 主机，这个 DNS 主要的功能之一，就是将主机名称转译成为 IP ，我想，这里您应该也已经了解了，对吧！如果是『不了解』，那么不要往下看了，请前往 简易 DNS 服务器 去瞧一瞧，瞧完了再回来继续吧！OK！好了，既然如此的话，那么使用邮件主机来寄信，并且不想要背主机所在的 IP ，那蛣M就一定需要让你的主机名称可以经由 DNS 系统来找到你的 IP 啰！对吧！没错，如果你真的要提供一个 Internet 上面的邮件主机，最好还是注册一个合法的主机名称，比较好记忆ㄋㄟ～

好了，接下来要讨论的就是，既然我的主机需要 DNS 来转译主机名称使成为 IP ，那么我真的就得必须要架设 DNS 吗？当然不是！要注意的是，我们刚刚提到的是『我就得在 Internet 上面注册一个合法的主机名称来对应 IP 』而不是『一定得要架设 DNS 在我的主机上面！』这个很重要，因为有太多的新手被 mail server 与 DNS server 的关系搞错乱了！如果到这里又混乱了！那么请，真的，一定，回到DNS 服务器那篇去慢慢的再从头读一次，否则.....也就是说，我们需要的是『合法注册过的主机名称』就是了！所以，你可以使用动态 IP 去申请一个动态 IP 的领域名称，也可以使用各大 ISP 提供的各项功能来注册，反正只要能够注册一个领域名称就是了！当然，你也可以自行去注册一个 DNS 主机，并且在你的主机上面建立 DNS 系统，但这并非是必要的！

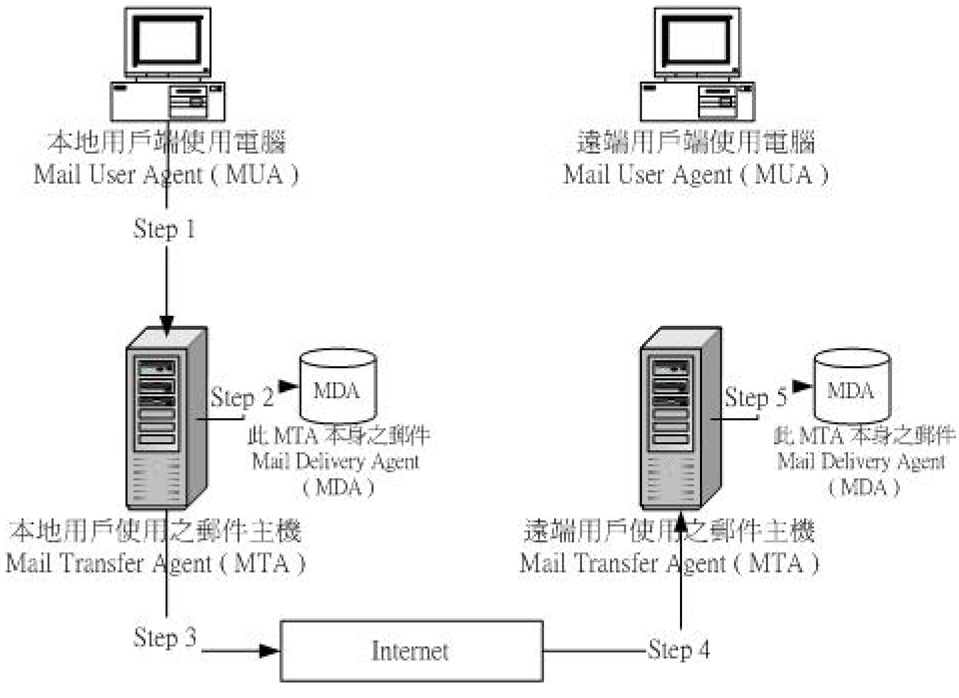
那么，假设我的主机名称对应 IP 已经成功的在 Internet 上面完成合法注册了，这样就好了吗？是这样没错啦！确实，只要有主机名称对应到 IP ，亦即是有 A (Address) 这个 DNS 的标志后，那么就可以架设 mail server 了，并且，一般来说，应该不会有问题的！然而， DNS 系统本身还有其它的功能可以支持 mail server ，使 mail server 更稳定与具有更佳的避免信件遗失功能，所以，就有 MX 这个 DNS 的标志产生啦！MX 这个 DNS 设定中的标志，主要就是要给 mail server 用的，基本上， MX 就是 Mail eXchanger 的缩写，他可以让 Internet 上面的信件马上找寻到 Mail 主机的位置，此外，由于 MX 后面可以接数字，因次，一个 domain 或者是一部主机，可以有多个 MX 标志，这有什么好处呢？主要的好处就是可以让，当主要的 mail server 挂点时，由于有 mx 标号，因此，信件不会直接退回，而是跑到下一个 MX 设定的主机去，并且暂存在该处，等到主要的 mail server 起来之后，这个 MX 设定的主机就会将信件给他传送到目的地！如此一来，甚至可以达到异地备援的功效呢！不只如此喔！MX 的功效还很多！最大的优点就是有点类似 router 的功能，我们或许可以称之为 邮件路由 吧！当有了 MX 标志之后，由于这是 DNS 的设定，所以当你要传送 mail 的时候，那么就可以直接依据 DNS 的 MX 标志直接将信件传送到该设定的 mx 邮件主机，而不需要去寻问到底邮件要寄到哪里去！这功能相当的不错的！因为可以让你的邮件很快的而且正确的送达到目的地呢！此外，由于可以设定多个 mx ，因此，假设『此路不通』，也就是先使用的 mx 邮件主机不通的时候，那么信件就会往下一个 mx 邮件主机传送！这样可以避免信件被

退信的机会！当然就更加的稳定啰！不过，这里也要特别强调， MX 『一定』要设定正确，否则，呵呵！反而会让你的信件永远在 Internet 上面流浪呢！

一般来说，邮件地址的写法为： account@server.name 的写法，在小老鼠 (@) 前面的指的是『账号』，至于 @ 后面的则是主机的名称！当你寄出这样的一封信时，首先，你的邮件主机会先去 DNS 系统寻找 server.name 这个主机名称对应的 IP 与 MX 标志，若有 mx 标志，那么这封 e-mail 将会把信先送到该 mx 主机，然后再由该 mx 主机将信件送达目的地 (就是 server.name 这个主机啦) ，而如果有多个 mx 标志时，那么这封 e-mail 会送到最优先的 mx 主机去(也有可能这部主机就是目的地主机喔！)，然后交给该主机来处理啰！而如果没有 mx 标志的话，那么在查得 IP 之后，信件才会慢慢的送达该邮件主机啰！在送达到邮件主机后，该主机则以前面的『账号』将信件发送到各个使用者的邮件目录下！所以啰，为什么说 mail 与 DNS 系统相关性很高呢？嘿嘿！由上面的说明您应该就不难了解啦！ ^_^

邮件的传送流程、MUA、MTA、MDA

约略了解了 DNS 与 mail server 之间的关系之后，在接下来我们要了解的是，那么 mail 到底是如何传送到目的邮件主机的呢？底下我们分成『寄信』与『收信』两个主要的邮件主机使用方式来加以介绍啰！先说明一下关于『寄信』的部分好了，通常我们都是使用桌上型计算机来寄信的，举个例子来说好了，如果你以 Netscape 或者 Kmail 或者 OutLook Express 来寄信的时候，那么那封信到底是怎么送出去的呢？可以参考一下底下的图示来说明：



图一、电子邮件以邮件主机寄送信件示意图

先来说明一下什么是 MUA, MTA 与 MDA 什么的，再来说信件怎么传送的好了！

- MUA (Mail User Agent)：顾名思义， MUA 就是『邮件使用者代理人』，华特(what)？邮件还需要代理人，怎么回事呢？喔！这是由于通常我们 Client 端的计算机都无法直接寄信的(不然干嘛要邮件主机？)，所以，需要透过 MUA 来帮我们传达信件，不论是送信还是收信，Client 端的用户都需要透过各个操作系统提供的 MUA 才能够使用邮件系统。举个例子来说， Windows 里面的 OutLook Express, Netscape 里面的 mail 功能

与 KDE 里面的 Kmail 都是 MUA 啦！ MUA 主要的功能就是收受邮件主机的电子邮件，以及提供使用者浏览与编写邮件的功能！

- MTA (Mail Transfer Agent) ： MUA 是用在 Client 端上面的软件，那么这个 MTA 就是用在邮件主机上面的软件啦！他也是主要的邮件服务器喔！这个 MTA 就是『邮件传送代理人』的意思。也来顾名思义一下，既然是『传送代理人』，那么使用者寄出的信，与使用者要收信时，就是找他 (MTA) 就对啦！因为他要负责帮我们使用者传送嘛！没错！基本上， MTA 的功能有这些：

    1. 收受外部主机寄来的信件：既然是邮件主机，那么『接收信件』想必就是主要的功能啰！呵呵，答对了！所以啰， MTA 最主要的功能就是收受外部来的信件，只要这个信件里面有 MTA 内部的账号时，那么这封信就会被 MTA 收下来；
    2. 帮使用者传送 (寄出) 信件：既然可以收信，那么自然也就可以发信啰！没错啦！只要使用者具有合法的使用 MTA 的权力，那么该使用者就可以利用这部 MTA 将他把信传送出去！不过需要注意的是， MTA 会将信件送给目的地的 MTA 而不是目的地的 MUA 喔！不要搞错了！(注：曾经有个朋友跟我说，要我传数据给他，而因为他要接收我的信件，所以他的计算机 "指的是 Windows 那个 Client 端的计算机" 得一直开着,真是不方便!听到这句话时,害我吓了一跳～这个观念是不对的～因为使用者使用的是 MUA ，而信件『仅会送达到 MTA 主机上面』而已，收、发信件时，都需要透过 MTA 来帮忙处理的！所以，使用者在使用邮件编辑器"MUA"将数据编辑完毕之后，按下送出，并且成功的送到 MTA 之后，接下来的事情就是 MTA 的工作了，跟使用者的 Client 端这部计算机 "一点关系也没有了")
    3. 让使用者自己的信可以收回去：使用者可以将放置在邮件主机的信件收到自己的个人计算机上面收看。

  大致的功能就是这些啦！通常我们所说的 Mail server (邮件服务器) 就是指 MTA 而言的！

- MDA (Mail Delivery Agent) ： 『邮件递送代理人』主要的功能就是将 MTA 所收受的信件，依照信件的流向 (送到哪里去) 来将该信件放置到本机账户下的邮件档案中 (Mailbox)！或者是再经由 MTA 将这个信件送到下个 MTA 去！而如果信件的流向是到本机当中时，这个邮件代理人的功能可不止是将由 MTA 传来的邮件放置到每个使用者的 Mailbox 而已，他还可以具有邮件分析 (filtering) 与其它相关的功能呢！这个功能很了不起喔！怎么说呢？具两个例子来说好了：

    1. 如果你知道某个广告信件的主旨都是固定的，例如『AV 情色 XXX』，你想将这种信件直接给他丢掉垃圾桶,可以吗？当然可以啰!透过 MDA 邮件分析的功能，就可以将信件丢弃啦！
    2. 如果有一天你要出差去，看样子可能一个星期碰不到电子邮件了，但是你又不想让一些朋友认为你在耍大牌都不回信的....这个时候你就可以利用 MDA 的功能，让邮件主机分析到，当要送给你这个使用者的账号的信出现时，就自动回复一封回信,让寄件者知道你在忙碌中....呵呵!这样的功能是否很不错呢？还不止这样喔！其它的等一下后面再提吧！

- Mailbox ：『邮件信箱』说穿了，就是在你主机上面的一个目录下的，某个人『专用』的信件收受档案啦！举个例子来说，系统管理员 root ，在预设的情况下，他会有个信箱，预设的档案是在 /var/spool/mail/root 这个档案就是了，一个账号都会有一个自己的信箱喔！然后，当 MTA 收到 root 的信时，就会将该封信件存到 /var/spool/mail/root 这个档案中啰！使用者可以透过程序来将这个档案里面的信件数据读取回去喔！

好了，了解了 MUA, MTA 与 MDA 之后，再来说到那么如何将信寄出去呢？可以分为底下几个步骤：

1. Step 1 使用者利用 MUA 寄信到 MTA 上面：通常我们使用 MUA（ 例如 Outlook express ）写信的时候，你总是要定义出几个咚咚：
   - 发信人与发信网站：对啦，总是要有这个信息才行的嘛！这个发信网站就是等一下 Step 2 接收信件的那个 MTA 啦；
   - 收信人与收信网站：是的，就是 account@e-mai.server 的样式啦！那个 account 就是该 e-mail.server 里面的账号啦！

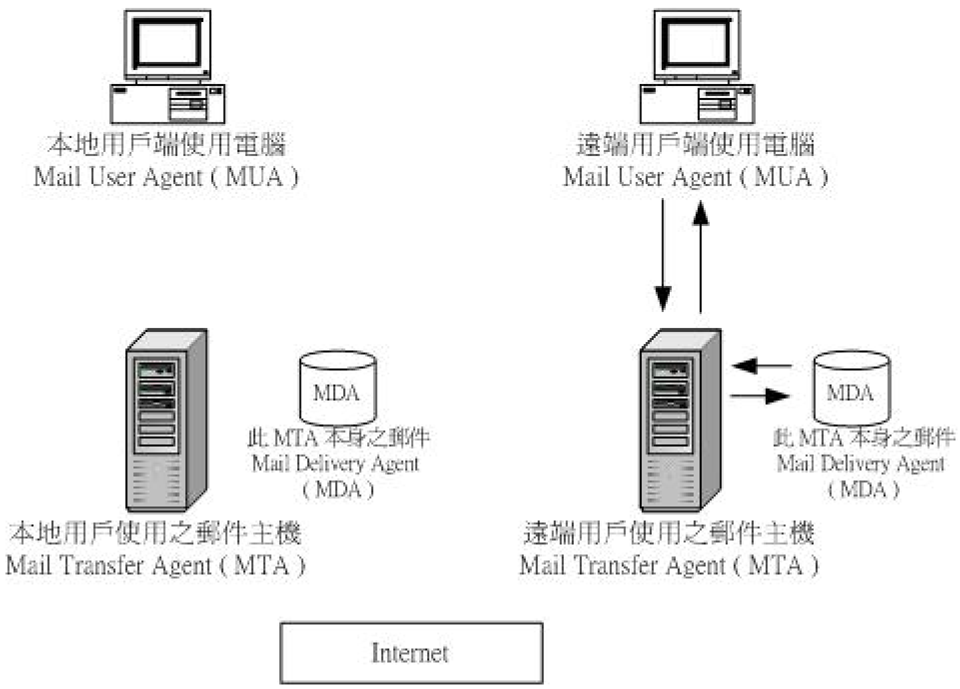
   好了，你在图一左上角的那部机器上面，也就是『本地端用户使用计算机』利用 MUA 的功能 （ 例如 Outlook express 好了 ） 写好了信之后，按下 MUA 的那个『传送』的按键，MUA 就会依据你所定义的主机地址将信发送到 MTA 上面；

2. Step 2 MTA 收到自己的信件，交由 MDA 发送到该账号的 MailBox 当中：如果在 Step 1 所收到的信件中，那个 e-mail.server 就是 MTA 自己，此时 MTA 会将该信件交由 MDA 去处理，将信件放置在收进者的信箱中；

3. Step 3 MTA 将信再转送出去：如果由 Step 1 来的信件的收件人并不是 MTA 的内部账号，那么该封信将会被再转送出去！由 Step 1 及 Step 3 的动作，我们也称为 Relay (邮件转递) 的功能喔！

4. Step 4 远程 MTA 收受本地的 MTA 所发出的邮件：远程的 MTA 会收受我们这部 MTA 的信件，并将该信件交给他的 MDA 来处理 （ Step 5 ） ，此时，信件会存放在远程的 MTA 上面，等待使用者登录读取或者下载回去！

整个流程大致上就是这样。这个时候，你由左上角的 MUA 将信件寄出之后，最后信件将会存放在右边那部 MTA 主机里面喔！还没有到达你的朋友的计算机 （ 就是右边的 MUA 那部计算机 ） ！这个时候，就要继续谈到收信的动作了！收信的动作有点像这样：



图二、客户端收受邮件主机的电子邮件示意图

远程用户使用的计算机直接连接到他的 MTA ，跟 MTA 要求察看自己的 mailbox 是否有信件，而 MTA 透过 MDA 去检查之后，如果有信件的话，就会将他传送回使用者的 MUA 中！同时，根据 MUA 的不同设定，MTA 会选择将该 mailbox 清除掉，或者继续保留！若继续保留的话，那么下次使用者再次的接收信件时，保留的信件会再次的被下载，因此，通常使用者 MUA 都是预设删除掉 MTA 上面的 Mailbox 内容的！接下来我们得谈一谈，那么寄信与收信使用的是什么协议呢？

使用的协议

总是得了解一下使用的协议呐!我们在寄信的时候,亦即由 MUA 将信件发送到 MTA 的过程中,以及 MTA 将信转递到下一个 MTA 的功能,目前绝大部分的邮件主机都是使用 SMTP (Simple Mail Transfer Protocol) 这个协议，port number 为 25 啦！在寄信的时候，你的 MUA 会主动的连接 MTA 的 port 25 ，然后将信经由 MTA 的 smtp 协议 （ port 25 ） 而送出去！而邮件主机 MTA 在转递的时候，也是经由下一部 MTA 的 port 25 来将信送出去的!所以啰,不论你是使用什么 MUA 或 MTA 邮件架设软件,只要大家都支持 smtp ，那么信件就可以顺利的流传啰!

收信呢？收信则是 MUA 经由 POP （ Post Office Protocol ） 协议来连接到 MTA 的使用者 Mailbox，以读取或者下载使用者在 Mailbox 当中的信件。，目前常用的 POP 协议为 POP3 （ Post Office Protocol version 3 ），这个协议产生的 port number 为 110 ，所以，你的 MUA 经由 MTA 的 port 110 将信件由 MTA 的 mailbox 当中将信件收到本地端的 MUA 上面供你浏览！同样的，只要 MTA 与 MUA 同时支持 POP3 这个协议，那么信件就可以自由的收受了！此外，目前也很流行使用 IMAP 这个协议来收受信件。在 pop3 的收信协议中，一般来说，当 client 端收完了主机端的信件之后，则该信件会主动的被主机端所删除！不过， IMAP 则可以避免这个问题！ IMAP 具有让使用者 （ client 客户端 ） 自行定义信件放置的目录功能，以及是否要储存下载的信件之后，原信件是否保留在主机上面的功能！目前我们常见的 Web 接口的电子邮件使用，大部分就是以 imap 来达成的！

所以我们知道了！通常一部提供收发信件的 MTA （ 不考虑 Web 接口的邮件主机 ） 至少需要两个协议，分别是 SMTP 与 POP3 ！而且，只要你的 MUA 与 MTA 同时均支持 SMTP 与 POP3 ，那么彼此就可以沟通啰！这也是为什么你使用 Outlook express 寄出的信，但是你的朋友可以使用 Netscape 收下来的原因！总之，

就是『网络协议』的沟通啦！

什么是 Relay 与认证机制

由图一的寄信流程图里面的第三步骤 (step 3) 中，我们知道， MTA 在分析收到的邮件之后，如果收件者不是本身主机的账号，则会将该信件再传送到下一个 MTA 上面，这个由 MTA 帮忙转信的功能就称为 Relay 啦。那么在这个功能当中，您有没有发现一件奇怪的事情啊！那就是：『是否任何人都可以使用我的 MTA 邮件主机服务器来传送他的邮件呢？』这个问题涉及到 Mail Server 的设定技巧了！如果设定不良的话，例如早期的 Sendmail 版本中，他就没有针对使用者来进行管制，也就是说，任何人都可以使用这样的一部邮件主机来达到信件传送的目的！这种主机我们称为『Open Relay』的电子邮件主机喔！这里请仔细的思考一下，如果我的 MTA 对于寄信的人没有一个限制的话，结果会如何呢？呵呵！没有错，结果就是任何人都可以使用你的 MTA 来发信了！那有什么好可怕的？我们在前言的地方就已经稍微说过了，那个所谓的『广告信、垃圾信件』的问题，而如果你的 MTA 没有对寄信的人作限制的话，由于任何人都可以使用你的 MTA 来发信，你的 MTA 将会变的『很笨重！』什么意思？那就是，你的 MTA 将会帮任何人寄信，如此一来，你的『网络频宽将会被广告信件所用光！』结果将导致你的 MTA 变成『Open Relay 主机黑名单的一份子～』！！！

为了避免这个问题，所以，目前所有新版的邮件主机服务器架设软件 (Mail server packages) 预设的情况之下，都不会对外完全的开放 Relay 的功能的！预设通常仅『针对主机 (localhost) 开放 Relay 的功能！』，不过，这样的 MTA 是可以收受来自 Internet 上面的，注明收件者是我们 MTA 主机内部账号的信件的，因此， MTA 在『收信』上面是没有问题的！

但是关闭了 Relay 之后，虽然可以避免掉我们 MTA 主机被当成广告信发送站，不过如此一来又造成了一些困扰！何解？因为通常我们仅针对主机，或者一些规范的 IP 或者是网段等信任的主机来开放他们的 Relay 的功能，所以在这个设定的范围内的 Client 端计算机可以自由的收发信件，至于没有规范到的 IP 来源的寄信信件，将完全的挡掉。然而万一您使用的是 ADSL 计时制的呢？又或者您是常常在外面出差的大老板，则你的 IP 将『不会固定』，完蛋啦～怎么办？既不能完全开放 Relay ，又没有固定 IP ，无解了吗？呵呵！还好，有所谓的 邮件认证机制 来帮我们解决这个困境啦！

所谓的『邮件认证机制』就是在刚刚我们图一的寄信流程图中，在 MTA 当中加入需要检查发信者的『账号与密码』比对的功能，当 MTA 接到来自 Client 端的传信需求时，会检查来自 Client 端的认证比对(账号密码)，如果账号与密码比对正确，则开始接受信件并帮忙转信，如果比对不正确则将该 MTA 并不会接受该封信件，直接在 Client 端显示『不接受您的信件』之类的讯息喔！目前有相当多种的邮件认证机制，这里我们偏向于介绍目前广为使用的 SMTP 邮件认证这个机制。

所谓的 SMTP 邮件认证机制，顾名思义，就是在 smtp 这个协定上面动手脚的一个机制啰！亦即是在寄信的时候，(由 MUA 到 MTA 那个 step 1 的步骤中)，我们的 MTA 主机『一定要求检验 MUA 发信者的账号与密码!』这样的功能!果真能做到这一点的话，那么你的 MTA 就可以在经过认证之后，提供认证者的 Relay 功能，而不需要针对某些信任网域或 IP 来分别设定开放 Relay 的功能啦！因为经由『认证』的机制，你的 MTA 会去分析寄信者的相关信息，通过后才会接受信件并帮他们寄信，否则就不接受信件！呵呵！没错！就是这样！透过这样的机制，您将不需要规范 Relay 的 IP 或网段，直接交给 SMTP 邮件认证来帮你管理你寄件者的 Relay 功能，从此以后，你的 Clients 就不会常常向你抱怨说 MTA 不稳定啰！

我们底下将介绍使用 cyrus-sasl 这种密码验证的认证机制啰！好了！底下我们将要介绍一下目前邮件服务器占有率上面应该依然是第一的 sendmail 这个 mail server 的架设！

套件安装：
使用 RPM 来安装 Sendmail 实在是『快乐得不得了～』太简单了～目前提供 Sendmail 做为邮件主机服务器的主要为 Red Hat 这个 Linux distribution ，至于其它的 Linux distribution 是否提供 Sendmail 就得请您自行到该官方网站上面查询一下啰！底下我们主要是以 Red Hat 7.x 以及 Red Hat 9 的 Linux 系统做为 Sendmail 的介绍，此外， OpenLinux server 3.1.1 亦是使用此一相关功能套件的喔！那么需要安装哪些套件呢？还记得我们在 Mail Server 使用的协议里面谈到的几个基本的协定吧？亦即是 SMTP 与 POP3 这两个，此外，由于 Sendmail 必须『读入』一些数据库格式，所以也必须要安装相关的数据库的函式库喔！

不过，如果您的系统是比较老旧的，例如 Red Hat 6.x 以前的版本，又或者是您的系统本来就不存在 Sendmail ，例如 Mandrake 等其它版本的 Linux distribution 时，那么您就得使用 Tarball 的方式来安装了！（ 事实上，几乎所有的 Linux distribution 都会纳入 sendmail ，只是有些套件，例如 Mandrake 预设是安装 postfix 就是了！ ）安装 Tarball 的 Sendmail 真是一件很『雪特』的苦差事，而且安装的不够好的话，还有可能产生一些设定上的困扰，此外，安装的过程当中，使用到很多的『天书一般的设定档案与牛屎一般的一大沱设定数据』，这些数据如果没有一定程度的 Sendmail 架构知识，是无法安装起来的，还有还有， Tarball 安装的话，最好是所有的 Sendmail 相关套件都一起安装，而不是分开来安装，所以，鸟哥将 Sendmail 的 Tarball 安装方法放在最后面，希望您至少看完『主机的设定』该节的完整内容，以及浏览过『关于邮件主机安全的设定』之后，再来尝试以 Tarball 完整的安装起属于您自己的 Sendmail 邮件主机喔！

好了，底下我们就来安装 Sendmail 及 POP3 这两个邮件服务器上面的组件吧！

使用 RPM 安装 sendmail （ 适用于原本 Linux 就是使用 sendmail ）
如果您是使用 Red Hat 7.3 以前的版本，例如 Red Hat 7.1, 7.2, 7.3 ，或者是 Open Linux Server 3.1.1 的话，那么请先确定一下底下的套件是否已经安装上去了呢？

```
[root@test root]# rpm -qa | grep sendmail
sendmail-cf-8.11.6-3
sendmail-8.11.6-3
# 若有属性相依的问题时，请将您的原版安装光盘拿出来， mount 上去后，
# 仔细的，一个一个的将相依的套件安装上去啰！ ^_^
[root@test root]# rpm -qa | grep m4
m4-1.4.1-5
[root@test root]# rpm -q mailx
mailx-8.1.1-22
```

那个 sendmail 就是主要的邮件服务器程序， sendmail-cf 是一些设定档案， 这两个套件是『一定』要安装的！至于那个 m4 的套件，则是转换 sendmail 设定文件的一支程序啰！也要安装喔！而那个 mailx 就是提供最简单的 mail 这支寄信与收信的套件啦！由于我的测试系统是 Red Hat 7.2 ，所以使用的算是比较旧一点点的 sendmail 8.11.6 版，如果您想要换装新版的 sendmail 8.12.xx 的话，请参考底下『Tarball 完整安装 Sendmail 服务器』的步骤！不过，这里我们先还是玩一玩这个预设的版本即可！记得喔！安装

完毕之后，请到 Red Hat 的网站上面去下载更新的 RPM 来更新吧
http://www.redhat.com/apps/support/errata/！或者是台湾的映射站喔！
ftp://linux.sinica.edu.tw/pub1/redhat/updates/

使用 RPM 安装 IMAP 套件
这个 IMAP 套件，就是负责收信的 POP3 那个协定啦！请使用 RPM 确认他已经安装在您的系统上面啰！

```
[root@test root]# rpm -qa | grep imap
imap-devel-2001a-1.72.0
imap-2001a-1.72.0
```

那个 imap 就是我们主要的 POP3 那个协议的套件啰！如果您是使用 Open Linux 的话，那么设定档应该会是在 /etc/inet.d 内，而如果是使用新的 xinetd 的话，那么设定档就会是在 /etc/xinetd.d 里面啰！等一下我们再来好好的谈一谈啦！（ 注：本章节并没有谈到 imap 这个协议的设定与应用，事实上， imap 这个套件同时提供了 pop2, pop3, imap 等协议的设定与相关功能喔！ ）

主机的设定：
知道如何安装 Sendmail 之后，接下来，我们得了解一下在邮件服务器架设之前，您需要先进行什么样的工作呢？

- 若想要架设的邮件主机未来是对 Internet 提供服务的，那么请确定您已经申请了『主机名称』或者已经具备有『经过合法授权的 DNS 主机』的服务了！重要的地方在于你的主机必须能够让大家在 Internet 上面查询的到啊！
- 虽然有 A 这个 DNS 的标志就可以架设 Mail server ，不过，毕竟有 MX 标志还是比较好的，所以，特别提醒大家，如果要架设 Mail Server ，最好(非必要)还是请您的上层 DNS 主机帮您设定 MX 标志，或者，您自己拥有 DNS 主机管理权时，可以自行设定 MX 这个标号才好！

好了，既然是玩 Sendmail ，那么就得了解一下 sendmail 的相关档案与说明啰！

Sendmail Server 的档案架构与基础说明

Sendmail 几乎所有的设定档都安置在 /etc/mail 底下，不过，如果你是以 RPM 安装的话，那么还有所谓的 sendmail-cf 的设定档，这个就是使用 M4 在进行 sendmail.cf 设定的程序！由于 Sendmail Server 所使用到的套件并不少，这包括有 sendmail, imap 以及 m4 等等，我们针对这些套件来谈一谈每个目录与档案下的数据吧！

- 设定档：
  Sendmail 的设定档几乎全部都在 /etc/mail 底下，但是也不一定！因为还需要看当初你建立 sendmail.cf 这个主要设定档时，将各个档案放置的地点而定！这部份可以使用 RPM 的方式来反查出你的设定档案的路径。Sendmail 与相关套件的设定档与相关的说明为：

- /etc/mail/sendmail.cf 或 /etc/sendmail.cf：这个就是 sendmail 的主要设定档，所有的参数都是他在管理的！但是，这个档案内的各个设定被号称为『天书』，所谓的天书就是『非一般人看的懂得！』，就连 sendmai 官方网站自行开发出来的设定程序也都『告诫大家不要手动编辑这个档案』，所以这里我们也不谈这个档案的内容啦！但是既然这个是主要设定档，那么又不要让大家手动编辑，那我要怎样进行 sendmail 设定的修改呢？这个时候就需要使用到 M4 这个指令了！ m4 可以将简单的一些环境设定参数，重新以内定的函式库或者函式定义来『制作』 sendmail.cf 这个设定档呢！sendmail 预设的 sendmail.cf 放置在 /etc/mail/sendmail.cf ，但是某些 Linux distributions 则将他改放在 /etc/sendmail.cf 这里～

- /usr/share/sendmail-cf/cf/xxxx.m4 ：刚刚我们提过那个 sendmail.cf 对吧！而由于这个档案最好不要手动修改，所以需要使用到 m4 这支程序。 m4 可以将一个简单的环境设定档转成 sendmail.cf ，那个环境设定档就是 sendmail-cf 这个套件所提供的啦。在 Red Hat 7.x 的系统中，主要的环境设定档就是 /usr/share/sendmail-cf/cf/redhat.mc 这个档案喔！不过，在 Red Hat 7.3 以后的所有 Red Hat Linux 版本当中，这个档案被移动到 /etc/mail/sendmail.mc 了！至于其它的 Linux 版本则请参考你的 sendmail-cf 套件的内容！

- /etc/mail/local-host-names ：这个档案主要用来处理一个主机同时拥有多个主机名称时候的收发信件主机名称问题。这个档案的用途可大了！当你的主机拥有多个 HOSTNAME 的时候，例如我的主机拥有三、四个主机名称，那么是否每个名称都可以用来做为收受信件的主机名称 (To: ..) 呢？并非如此！如果你的主机名称为 test1.your.domain 以及 test2.your.domain ，而且这两个 hostname 您都希望可以用在收受电子邮件，果真如此，那么，你就必需将这两个名字都写入 local-host-names 这个档案当中，一个主机名字占用一行。注意：没有写入这个档案的『你的主机名称』，那信件将无法正确的寄达这部主机喔～例如：www.vbird.adsldns.org、vbird.adsldns.org 这两个主机名称的 ip 都是相同的，也就是指向同一台机器上。假设这台主机名称预设为 vbird.adsldns.org，那在预设情况下，寄给 userid@vbird.adsldns.org 都是 ok 没有问题的！但是寄给 userid@www.vbird.adsldns.org 就会出现错误。其中原因是因为没有告诉 MTA 除了 vbird.adsldns.org 这个主机名称外，还有 www.vbird.adsldns.org 也是指向这台主机上。所以寄给 userid@www.vbird.adsldns.org 会出现错误，通常就是 mail loop to me，要不然就是不允许 relay 的错误情况。

- /etc/mail/access.db ：这个是『规定谁可以或不可以使用本邮件服务器的数据库』，要转成这个数据库需要藉由 makemap 以及 /etc/mail/access 档案的配合！这个档案可以说是 Sendmail 里面最重要的『使用者权限管理』的数据了！在后面我们会继续说明。

- /etc/mail/aliases.db 或 /etc/aliases.db ：这个 aliases.db 是用来设定『信箱别名』的一个咚咚！你可以藉由这个档案的设定来规范你的『群组收信』喔！不过，还需要藉由 aliases 及 newaliases 来做成这个档案才行！

- /etc/mail/statistics ：这个档案在记录 Sendmail 收发信件的相关信息喔！

- 执行档：
  Sendmail 的执行档也不少，得说一说：

  - /usr/sbin/sendmail：就是 sendmail 的主要执行档啦！他会读取 sendmail.cf 这个档案的设定内容喔。你在发送信件时，就是使用这支程序啦！启用这支程序之后，预设的启用的 port 是 25 咯。

  - /usr/sbin/ipop3d：sendmail 的功能是在处理寄信问题，而 ipop3d 就是处理 client 的收信问题啦！如果你的 Mail Server 希望提供客户端使用 Netscape 或 OutLook express 来收信，那么就需要提供这个服务才行！这个服务的设定档在 Red Hat 当中是在 /etc/xinetd.d/ipop3 ，如果是 Open Linux server 3.1.1 的话，那就会变成在 /etc/inet.d/imap 这个档案中。注意： pop3 是由 imap 套件所提供的，并没有包含在 sendmail 套件之中喔！

  - /usr/sbin/makemap：主要将 access 转成 access.db 的数据库制作的执行文件；

  - /usr/sbin/mailstats：将 /etc/mail/statistics 档案读出来的一支程序！可以查看到目前为止 Sendmail 工作共传送、接收多少邮件啰！

  - /usr/bin/newaliases：将 /etc/mail/aliases 转成 /etc/mail/aliases.db 的执行档！

  - /usr/bin/mailq：用来观察 /var/spool/mqueue 这个邮件暂存目录的数据情况的指令！

  - /usr/bin/m4：这个就是将 *.mc 档案转成 *.cf 档案的主要执行档啰！需要搭配 sendmail 原始码，或者是 sendmail-cf 这个套件才行！注意： m4 是也需要额外的安装的一个套件喔！sendmail 原本套件中并未包含 m4 这个套件！

- 邮件相关目录：
  sendmail 接收下来的邮件放置在哪里呢？

  - /var/spool/mail ：这个是邮件『收受下来之后，每个使用者信件放置的目录』，一个账号会使用掉一个档案，例如你的账号为 test ，那么你的信在 Server 中时，就是 /var/spool/mail/test 这个档案了！此外，你的 POP3 的协议亦是使用这个目录中的 mailbox 做为预设的邮件取得的档案数据。

  - /var/spool/mqueue：当邮件由于对方主机的问题，或者是网络的问题，而无法送出去时，那么该封邮件将会暂时的存放在这个目录下，然后主机会每隔大约 30 ~ 60 分钟重新尝试传送一遍，通常设定在五天内该封信件还寄不出去，那就会退给原发信者了！

  - /var/spool/clientmqueue：这是新的 sendmail 8.12 版本才会出现的队列目录（您如果想要以 tarball 安装 sendmail 的话，请务必参考本章底下的说明，这个目录的权限设定相当的重要喔！）。

  大致上的档案就是这样啦！接下来谈一下如何设定 sendmail 吧！

使用 m4 来简易设定 sendmail

一般来说，只要您在安装完了 sendmail 之后，您的 Mail Server 就可以正式的来启动了！但是不幸的是，在 Red Hat 7.xx 以后的版本中(包含 Red Hat 9)，为了杜绝广告信件的问题，所以在预设的情况下，您的 sendmail 将『只会监听 127.0.0.1 这个接口的收发信件需求！』至于非 Red Hat 版本的 sendmail 则可以正常的来启动喔！为了解决这个问题，所以我们势必要针对 sendmail.cf 这个设定档案来进行修订，但是这个档案原本即建议需要由 m4 来进行修改，所以，我们就得了解一下使用 m4 来转换成为 sendmail.cf 的设定档 file.mc 的相关参数啰！

- 建立 M4 参数档：
  m4 的参数档通常档名均取为 filename.mc 这样的附档名格式，你可以在 /usr/share/sendmail-cf/cf 里面找到相当多的范例档案喔！例如 Red Hat 的设定范例文件为 /usr/share/sendmail-cf/cf/redhat.mc（如果是 Red Hat 7.3 以后版本，含 Red Hat 9 ，这个档案则放置在 /etc/mail/sendmail.mc 喔！）。这个环境参数设定文件的设定项目很多，其格式为：

  ```
  设定组件(`设定项目', `参数一', `参数二')
  ```

仔细看到上面的例子当中，在设定的组件后面接上小括号，而小括号内则为该设定组件的项目内容，以及该项目内容的参数！而将设定项目与各参数包起来的『并不是单引号』，要注意的是，在『设定项目』左右两边的：

1. 左边的是 quod ，也就是键盘上面数字键 1 的左边那个按键『`』；
2. 右边的才是单引号『'』。

这里很容易被搞错！请特别注意，而每个设定项目与参数之间，则是以逗号『,』来做为分隔喔！底下我们谈一谈几个主要的设定组件与各个设定组件底下的设定项目吧！

- divert ：这个组件仅是在于提供『是否要将说明数据(或者是批注数据)写入输出的档案中』而已，如果在 filename.mc 档案当中具有批注符号时，(注意， *.mc 的批注符号可以是 # 也可以是 dnl 这个字符串！)而你输出资料时不想将这些说明资料也输出，那就可以使用 divert(-1) 。反之，如果你想将这些说明数据同时输出，那就使用 divert (0)。由于我们不想要手动修改 sendmail.cf ，所以输出的数据当然就不太需要注明啦！只要在环境设定档 *.mc 里面说明清楚即可!因此,你应该会常常在档案当中看到 diver (-1) 才对！范例为：

  ```
  divert (-1)
  ```

- OSTYPE：这个组件功能在设定使用的操作系统类别！ Sendmail 预设提供数种操作系统的模式,你可以在 /usr/share/sendmail-cf/ostype 这个目录当中找到所支持的操作系统模式。因为我们是使用 Linux ，所以范例为：

  ```
  OSTYPE(`linux')
  ```

- define：这个组件的作用比较多喔！他可以定义出许多有用的 sendmail 需要的参数，举个例子来说，如果我要将邮件别名设定档放置在 /etc/aliases 底下，那么我可以使用底下的范例：

  ```
  define(`ALIAS_FILE', `/etc/aliases')
  ```

  那个 ALIASE_FILE 就是主要的设定项目啦!而这个项目主要规定邮件者别名的档案所在地，所以啰，后面就直接接上完整的文件名称啦！更多详细的 define 说明，可以参考您计算机中的 /usr/share/sendmail-cf/README 这个档案喔！

- undefine：恰恰与 define 相反啦！ Sendmail 预设会支持定义很多的项目，而如果您不需要定义该项目，则可以使用 undefine 来将他移除掉喔！例如：

```
undefine(`UUCP_RELAY')
```

- FEATURE：这个组件 FEATURE 字面上的意思是『特征、特色』，那也就是说，这个组件里面会规定出 sendmail 所额外新增的一些任务啦！这些任务的支持必需要 sendmail 有提供才可以！你可以在 /usr/share/sendmail-cf/feature 这个目录当中找到 sendmail 所提供的各个功能喔！举个例子来说，如果我们要规定 sendmail 存取权限设定的档案，也就是 /etc/mail/access.db 时，你可以这样写：

```
FEATURE(`access_db',`hash -o /etc/mail/access.db')
```

  注意：上面 access_db 是某个任务的项目，而后面接的 hash 是数据库格式，至于 sendmail 所使用的数据库则是 /etc/mail/access.db ！更多的 FEATURE 相关设定项目可以参考 /usr/share/sendmail-cf/README

- MAILER：这个组件在设定所使用的邮件主机传送邮件(递送, delivery)的代理人，一般而言,我们的代理人都是 smpt 协议啊,不过,如果我们主机内的用户(主机 /etc/passwd 存在的实体用户)想要使用 sendmail 来寄信,那是否仍然要透过 smtp 这个代理人呢？不太需要的， sendmail 本身就提供发信的功能，而要让主机上面的实体用户可以在登入主机环境的中使用 sendmail ，那你就必需要启动 local 这个本地端的邮件递送功能啦！因此，通常这个组件会设定为：

```
MAILER(local)
MAILER(smtp)
```

  如此一来，当 sendmail 发现信件来自于主机内部，那就会使用 local 来传送信件，当信件来自于主机外部时,那才会使用 smpt 协议来寄信喔!未来还可以新增 procmail 这个 MDA 呢！ sendmail 支持的 MAILER 可以在 /usr/share/sendmail-cf/mailer 这个目录中查询的到！

各个设定组件我们先介绍到这里,至于更详细的说明,请务必到 /usr/share/sendmail-cf/README 这个档案当中寻找！至于其它更完整与新鲜的 M4 设定项目，请到 http://www.sendmail.org/m4/readme.html 查询喔!底下我们来聊一聊这个环境参数档设定完毕之后(或者称为 M4 scripts)，要怎样来『制作』sendmail.cf 呢？

* m4 程序的执行

m4 程序在执行的时候，必需要先读入相关的参数项目才行，这个参数项目在 /usr/share/sendmail-cf/m4/cf.m4 这个档案当中，因此，如果你的 *.mc 档案里面没有这一行：

```
include(`/usr/share/sendmail-cf/m4/cf.m4')
```

那么你就必需要执行两个档案的 m4 转换，否则只要执行一个即可！假设您的 m4 script 档名为 redhat.mc ，那么您可以这样转换 sendmail 所需要的 sendmail.cf ：

```
Red Hat 7.2 以前版本：
1. 若 redhat.mc 里面没有 include 的项目，则：
[root@test root]# cd /usr/share/sendmail-cf/cf
[root@test cf]# m4 /usr/share/sendmail-cf/m4/cf.m4 \
>  redhat.mc > redhat.cf

2. 若 redhat.mc 里面已经包含了 include 的项目，则：
[root@test cf]# m4 redhat.mc > redhat.cf

Red Hat 7.3 (含 Red Hat 9) 以后版本：
1. 若 sendmail.mc 里面没有 include 的项目，则：
[root@test root]# cd /etc/mail
[root@test mail]# m4 /usr/share/sendmail-cf/m4/cf.m4 \
>  sendmail.mc > redhat.cf

2. 若 sendmail.mc 里面已经包含了 include 的项目，则：
[root@test mail]# m4 sendmail.mc > redhat.cf
```

上面制作而成的 redhat.cf 就是 sendmail.cf 的内容啦！然后请将你的 sendmail.cf 备份，举个例子来说， Red Hat 的 sendmail.cf 在 /etc/ 底下，所以我可以这样做：

```
[root@test cf]# mv /etc/sendmail.cf /etc/sendmail.cf.old
[root@test cf]# cp redhat.cf  /etc/sendmail.cf
# 若是 Red Hat 7.3 以后版本，则
[root@test mail]# cp redhat.cf  /etc/mail/sendmail.cf
```

这样就成功啦！

- 修改 Red Hat 7.x 以后版本 (含 Red Hat 9)的设定档：

  我们说过，Red Hat 7.x 以后版本的设定档里面已经将邮件来源的接口定义为『仅来自 127.0.0.1 这个界面』，所以我们必需要开放监听的界面才行！请注意，这里仅开放『监听』而不是开放 Relay 喔！如果您使用 Red Hat 7.1/7.2 ，那么请修改您的 /usr/share/sendmail-cf/cf/redhat.mc ，如果是 Red Hat 7.3 ，则修改 /etc/mail/sendmail.mc ！我们这里都以 sendmail.mc 为主来说明，如果您是使用 Red Hat 7.2 以前版本，请记得搜寻相关的档案喔！至于非 Red Hat 系统，例如 Open Linux ，则可以略过这个步骤喔！

```
1. 寻找档案的内容
[root@test root]# cd /etc/mail
[root@test mail]# vi sendmail.mc
找到下面这一段：
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
将他改成
DAEMON_OPTIONS(`Port=smtp,Addr=0.0.0.0, Name=MTA')
储存后离开

2. 重新制作档案：
[root@test mail]# m4 sendmail.mc > redhat.cf
[root@test mail]# mv sendmail.cf sendmail.cf.old
[root@test mail]# cp redhat.cf sendmail.cf
```

-

  这样就大功告成啰！

启动 Mail Server

Mail Server 的启动是相当的简单的，在 Red Hat 的系统当中，你可以依序启动 sendmail 以及 POP3 这个服务喔：

```
1. 启动 sendmail
[root@test root]# /etc/rc.d/init.d/sendmail start
Starting sendmail:                                          [  OK  ]

2. 启动 POP3 这个协定
```

```
[root@test root]# cd /etc/xinetd.d
[root@test xinetd.d]# vi ipop3
# 找到下面这一行：
disable = yes
# 将他改成
disable = no
# 储存后离开！至于更详细的说明，可以参考『基础学习篇的认识服务』喔！

[root@test cf]# /etc/rc.d/init.d/xinetd restart
[root@test cf]# netstat  -tl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:pop3                  *:*                     LISTEN
tcp        0      0 *:smtp                  *:*                     LISTEN
```

看到了吗？这样我们的 Mail Serve 就已经启动了！他已经可以进行收信与送信的工作啰！事实上，如果您曾经以 vi 检查过 /etc/rc.d/init.d/sendmail 这个档案的话，你会发现他其实就是使用 /usr/sbin/sendmail 这支程序在工作的啦！他通常的指令下达方式为：

```
[root@test root]# /usr/sbin/sendmail -bd -q30m
参数说明：
-bd ： 表示将 sendmail 以 daemon (可以想成常驻内存的类型) 的类型启动！
-q  ： queue 的意思，后面接的是时间参数，时间参数有：
       s (秒) m(分) h(小时) 及 d(天)
       -q30m 表示每隔三十分钟，会将放置在邮件队列 ( 一般是在 /var/spool/mqueue )
       的邮件尝试寄出一次！所以，上面的例子当中，说的是将 sendmail 以 daemon
       的类型启动之后，并且每 30 分钟去将邮件队列(为寄出的邮件)尝试寄出一次！
```

而 Red Hat 针对邮件队列寄送邮件的时间，则是设定在 /etc/sysconfig/sendmail 这个档案里面喔！

注：
如果您在启动 sendmail 的时候，出现类似这段文字：
*** Warning: File `virtusertable.db' has modification time in the future (2003-01-13 11:57:26 > 2003-01-13 06:04:40)
make: warning: Clock skew detected. Your build may be incomplete.
这表示您在安装 Linux 的时候，可能发生一些时间上面的错误判断了！导致于你的 sendmail 相关的设定档竟然比目前的时间还要新，也就是该档案乃『来自未来』～这个时候你可以使用：
touch /etc/mail/*
来将档案的时间更新为目前的时间，那就可以顺利的启动 Sendmail 啰！ ^_^

设定主机名称

启动了 Sendmail 之后，还需要规定你的主机名称喔，这样，你的邮件主机才能正常无误的开始工作呐！

假设您的主机 IP 可以在 Internet 上面以 DNS 系统查询到，以我为例，我的机器上面有 www.tsai.adsldns.org 及 tsai.adsldns.org 两个主机名称，并且这两个主机名称均指向我家里的那一部机器，那么假如我只要 tsai.adsldns.org 具有收信的资格，如果人家寄信到 www.tsai.adsldns.org 则不予以接受，并将该封信件退回去时，可以这样做：

```
[root@test root]# cd /etc/mail
[root@test mail]# vi local-host-names
tsai.adsldns.org
```

在该档案里面仅存在一个主机名称即可！那么万一我有三个主机名称，分别是 tsai.adsldns.org, www.tsai.asdldns.org 及 tsai.linux-site.net，而且我这三个主机名称都希望可以收到信件时，那么你就必需要这样修改啰：

```
[root@test root]# cd /etc/mail
[root@test mail]# vi local-host-names
tsai.adsldns.org
www.tsai.adsldns.org
tsai.linux-site.net
```

每一行有一个主机名称！请记住，未来如果您的主机新增了不同的 HOSTNAME ，并且你希望该主机名称可以用来收发信件，那么要修改 local-host-names 喔！那么什么时候不要将所有的主机名称都给他写到这个 local-host-names 里面去呢？！由于目前的广告信件实在是太多了，他们会主动的在 Internet 上面搜寻一些主机名称，然后随机四处发送垃圾邮件。如果您的主机有四个名称，每个主机名称都写到 local-host-names 去的话，呵呵！很可能一封广告信您会『收到四次』，因为四个主机名称嘛！所以啰！有的时候还是不要让全部的主机名称都可以收信是比较好的呐！

设定邮件服务器使用权限 /etc/mail/access

在你启动了 Sendmail 以及设定好主机名称 (local-host-names)之后，那你就可以利用『在你的主机上面利用你的主机来寄信』了！为什么要加上『在你的主机上面』呢？还记得我们在前面有提过，为了杜绝广告与垃圾信件，所以预设的情况中， Sendmail 是关闭 Open Relay 的对吧！但是为了主机使用者的方便，所以我们预设是有启动 local 这个 Mailer ，也就是说：

1. 只有使用者是在主机上面发信的，例如使用 SSH 登入主机后，以 mail 这个指令来发信;
2. 又或者使用者直接在主机上面使用 X-Window System 里面的邮件代理人，亦即是 Netscape 或者是 Kmail 等软件来发信

的时候，我们的 Sendmail 才会帮使用者『寄信』喔！至于其它的计算机来源的『寄信』邮件，Sendmail 一概将他退信回去！那么如果我想要在其它的计算机上面使用我这部 Mail Server 来寄信呢？这个时候就要编辑『/etc/mail/access』这个信任网域设定的档案啦！假设一个例子好了：

- 我的 Sendmail 主机想要支持我内部网域的所有计算机来寄信，而我内部网域的计算机 IP 网段为 192.168.0.0/24 这一段;

- 还有，我另外有一个公共 IP (Public IP)为 140.116.44.125 ，也想要让他可以寄信；
- 此外，我发现 192.168.1.100 计算机使用者都会乱寄垃圾给我的邮件主机，所以我想要挡掉他；
- 那个 h8h.com 也有问题，我也要挡掉这个网域 (domain) 的来源；
- 更发现有个使用者叫做 test@testing.domain.name 也是色情广告信业者！

那么我可以怎样设定 Sendmail 的存取权限呢？

```
[root@test root]# vi /etc/mail/access
# 预设情况下有启用的 IP
localhost.localdomain          RELAY
localhost                      RELAY
127.0.0.1                      RELAY

# 想要开放权限的 IP 与网域
192.168.0                     RELAY
140.116.44.125                RELAY

# 挡掉的 IP、主机名称与 E-mail
h8h.com              DISCARD
192.168.1.100        DISCARD
test@testing.domain.name    REJECT
# 储存后离开

[root@test root]# makemap hash /etc/mail/access < /etc/mail/access
```

在 /etc/mail/access 当中，只要有加上 # 就是批注内容，所以不会被读入数据库当中！其实真正的 Sendmail 可以读得到的数据库是 /etc/mail/access.db 这个档案，不过，我们必需先在 /etc/mail/access 编辑完毕后，再以 makemap 这支程序将他改写成为数据库!所以，不要忘记 makemap 那个步骤喔！由上面的档案我们就可以知道啦， access 的语法为

规定的范围　　　　　　　　　　规定可以在 sendmail 上面的动作

IP/不完整 IP/主机名称/E-mail　 RELAY/DISCARD/REJECT

在『规定范围』与『规定可以在 sendmail 上面的动作』两项目之间最好以 <tab> 按键来隔开会比较正确！『规定的范围』还可以设定『来源』与『目的』喔！例如：

from:test@your-domain.com.tw　REJECT

to:blah@your-domain.com.tw　 REJECT

当信件想要使用我们的邮件主机来进行寄信或者是 Relay 的动作时，首先会传送 mail header 到我们的 mail server 上面，这也就是一些邮件的基本数据(如认证信息、来源 IP、目标的 MTA 等等)，但并不包含邮件的内容(例如信件本体、附件夹带等等的内容)。也就是说，一封邮件基本上可以分为两大部分，分别是 Header 与 Body ，Header 仅记录邮件基本信息，Body 才是真正的信件内容。当邮件想要进入 MTA 时，会先发送 Header 给 MTA ，MTA 判断这个 Header 的信息是可接受的之后，才后继续接受来源主机的邮件 Body 内容!如果这些基本讯息传送过来的主机信息包括在 /etc/mail/access.db 里面所记录到的主机(或 IP)范围时，就可以指定底下这些动作：

- RELAY：允许该来源主机所传送过来的邮件可以被接受，然后再进行 Relay 的动作。以上面的范例为例，则 192.168.0.0/24 来源的计算机所发送来我们 Mail Server 的邮件将会被接受喔！
- REJECT：若来源主机的主机名称或 IP 在 REJECT 的情况下，则我们 Mail Server 将不会接受对方的邮件内容(就是 body 部分)，『并且会回传一个错误或警告讯息给原发信端』喔！
- DISCARD：与 REJECT 相似，亦即关闭规定范围内的计算机主机的 RELAY 功能，不过， Sendmail 会直接将该信件『丢弃』而不会『退回』！

通常我们比较建议使用 DISCARD 啦！为什么呢？如果该 IP 来源传送的是广告信件，您又使用 REJECT 的话，那么两方面的邮件主机将会一再地进行 mail header 的传送，也是挺消耗频宽的！所以我们比较建议使用 DISCARD！再来，由于 access 不支持网域的写法，亦即
192.168.0.0/24
192.168.0.0/255.255.255.0
这种类型的网域写法并不能被使用的！所以，只能以 192.168.0 这种写法来达成整个网域的设定了！也就是说，access 里面大概只能支持 A/B/C Class 的网域，subnet 大概就无法达成了！只能一个一个 IP 的 Keyin 啰！编辑完这个档案之后，就可以让你的其它主机使用 Sendmail 的 Relay 功能啰！还不赖吧！

重要观念：一封信件的收受流程

OK！到目前为止的 Sendmail 主机设定而言，你的 Mail Server 应该已经能够应付一个小型的企业单位了，不过，为了让您能够更清楚的知道『我要如何控制我的 Mail Server 』，我们底下将会以 Sendmail 收受一封信件的流程，来介绍信件的传送方向，好让您更清楚的了解到你的 Mail Server 在干啥好事喔！一般而言，当你的 Sendmail 收到一封信件时，他是怎样判断这封信件要怎样传送的呢？我们先谈一谈如果 Sendmail 收到一封『非本机端送出的来信』时，他是怎样处理这封信件的？

1. 当 MTA 收到一封邮件，并且该邮件的『信件收件者』为 MTA 本身的用户账号时，此时将会以本机端 (local) 的收件规则来进行收件，如果 /etc/mail/access.db 没有针对来源 IP 或者 host 或者 e-mail 抵挡时，则该封信会被我们的 MTA 收下来，并且储存到 /var/spool/mail 里面。例如，当我的 tsai.adsldns.org 收到一封给 vbird@tsai.adsldns.org 的邮件，并且 /etc/mail/access.db 没有针对来源抵挡 (在不考虑 procmail 的情况下) ，那么我的 tsai.adsldns.org 这部主机，会立刻将该封邮件存放到 /var/spool/mail/vbird 里面去，而不必经由『认证』或者抵挡的机制。请注意，在这个情况中， Sendmail 并不会去检查送件者是否来自于信任网域喔 (只要 /etc/mail/access 没有挡到的主机或 IP 或其它的 e-mail 信息)！

2. 如果这封邮件的『信件收件者』并没有 MTA 本身的用户账号时，那么 MTA 会以 SMTP 这个外送规则来传信，此时 MTA 会开始去检查 /etc/mail/access.db 这个数据库里面，任何有关于送件者的 IP、E-mail 以及相关的动作等，如果该封邮件有相关的数据在 /etc/mail/access.db 里面时 (不论是 RELAY, REJECT 或是 DISCARD) 那么该封邮件就会依照 /etc/mail/access.db 里面指定的行为进行邮件的动作(可能是 RELAY 或 DISCARD 等等)！

3. 如果该封邮件经过上面两道手续后，仍然找不到任何有关的动作讯息，那么这封邮件将会退回给原发信者！

上面的信件收受行为是在没有 SMTP 邮件认证以及 procmail 这个 MDA 管理的时候所具备的动作！如果加入 SMTP 或者 procmail 之后，会变成怎样呢？呵呵！先不告诉你，待会继续往下看再说吧！

关于广告信的收受
很多人常常会发现这样的一件事，就是：『为什么有人利用我的 mail server 寄信给我？』举个例子来说，假如我的一部 Mail server 主机名称为 mta.domain.name ，而他上面有个实体邮件用户为 user@mta.domain.name ！这个 MTA 主机并没有对外开放 RELAY 的功能喔！但是有一天， user 这个人还是接到广告信了！更神奇的是，该封广告信的发信者为 someother@mta.domain.name ！怪怪！明明我的 mta.domain.name 就是没有 someother 这个用户，怎么还可以用我的主机寄信给我自己呢？ 好了，现在请仔细的参考一下上面的三个步骤，你会发现一件事情，就是『第一个步骤中，如果发现该封信的收件者有本机的账号时，且 /etc/mail/access 没有抵挡到该封信时，则该封信件就会被接收下来！』对啦！就是因为如此，因此，对方可以用你的 mail server 寄信给你！不过，还好的是，这样的情况中，该封广告信只会在你的 Mail Server 内传送，并不会寄出去外部的！注：因为要寄到外部去，就需要 RELAY 的功能啦！

上面提到的是关于『来自 MTA 外部的信件』时的处理动作，那么如果这封邮件是来自于『主机内部』的行为呢？例如：使用者以 SSH 登入后，使用 mail 这个指令来执行寄信的动作，又或者是直接在 MTA 这部主机上面的 X Window System 内的 Kmail 来发信呢？由于我们刚刚在设定 sendmail.mc 这个档案的时候，你会发现一句设定值：

MAILER(local)

或者是

Cwlocalhost.localdomain

这两个设定都代表『本机寄出的邮件可以不用经过 SMTP 的手续，将直接以 sendmail 的功能寄出』，这也就是说，无论如何，来自主机内部的信件都将被传送出去！这也是为什么有的时候明明你的 sendmail 没有正常的启动，但是在主机上面直接以 mail 这个指令却还是可以将邮件送出的原因啦！

例题：曾经有朋友发现一个有趣的现象，那就是他的 WWW 网站提供 CGI 的功能 (所谓的 CGI 指的是一些动态的网页内容，例如鸟哥的私房菜里面的留言版，这些功能很多是利用 perl 语言或者其它语言写成的程序喔！)，他所提供的 CGI 程序的功能可以帮助使用者寄信，后来发现很多人便藉由这个 CGI 的网络功能，使用他的 WWW 主机发送大量的广告信，他就很生气的将他自己的 WWW 主机的 Sendmail 关闭，也就是将 smtp 的 port (25) 关掉，以为这样就可以将广告信杜绝啦！但是，广告信却还是一直的发送！并没有停下来！您知道为什么吗？！ 答： 原因应该很简单吧！因为 WWW 在 Linux 本机上面跑，而管理员提供的的 CGI 是在 WWW 上面跑，也就是说，这个 CGI 本来就是利用 Linux 的 Sendmail 在传送邮件的，那既然 sendmail 本来就可以不需要透过 smtp 的 port 来传送邮件，自然你的广告信就还是可以自由的发送出去啰！

设定使用者别名 /etc/aliases

一、群组寄信的功能：

约略了解了 Sendmail 整体之后，目前你的 Sendmail 应该也可以顺利正常的运作了！不过，还有个重要的课题要来讨论，那就是关于『群组寄信』的问题啦！假设你是在学校单位里面，在这所学校里面的每个同学都有自己的账号，而学校的老师也都是使用同学的电子邮件来联络感情！不过，要记住一个班级 30~50 个同学的电子邮件地址实在不怎么好记，加上未来同学们毕业，新的同学又加进来，哇！岂不头疼～这个时候您可以帮助这些善良的老师们啦！就利用这个『使用者别名设定』的功能即可！怎么作呢？我们可以将一个班级取一个代号，例如预计 92 年毕业的 13 班，就称为 student9213 这样的账号，但是这个账号并非是实体用户喔！他仅是一个别名而已！基本上，他代表了 92 年毕业的 13 班的全体同学的电子邮件！这个功能可以透过编辑 /etc/mail/aliases 来达成喔！(注：这个档案不一定在 /etc/aliases ，有时会在 /etc/mail/aliases ，完全依照当时使用 filename.mc 定义时的路径而定的！) 这个 /etc/mail/aliases 的语法有点像这样：

在邮件上面的收件者账号：　　真实账号 1，真实账号 2，真实账号 3....

birdhouse:　　　　　　　　bird1，bird2，bird3，bird4

在上面的例子中，『真实账号 1...账号 3 中间的所有账号与账号之间都以逗号隔开而已！』你也可以在逗号后面接空格符，这是没有关系的！但是不能只接空格符而没有逗号喔，不然就会造成人名的误判！而 birdhouse 那一行就更清楚啦!当我寄出一封信给 birdhost@tsai.adsldns.org 时,在 tsai.adsldns.org 主机收到这封信之后，会将该封信复制成四封并分别寄给 bird1, bird2, bird3, 及 bird4 四个使用者，所以你只要记住 birdhouse 就行了！所以，如果用在上面学校单位的那个例子时，就可以这样进行：

```
1. 编辑别名设定档:
[root@test root]# vi /etc/aliases
新加入这一行在 aliases 的最底下:
student9213:  st001,st002,st003,st004.st005,st006,st007.....

2. 制作数据库 /etc/aliases.db
[root@test root]# newaliases
```

要注意的是，与 /etc/mail/access.db 相似的，我们 sendmail 读取的数据库格式其实是 /etc/mail/aliases.db 这个档案，所以当你编辑完成 /etc/mail/aliases 之后，记得一定要使用 newaliases 这个指令来将数据变成数据库喔！否则 sendmail 将不会读取到您刚刚修改完成的变动！这个群组寄信功能相当的不赖，如果你有四个计划在你的 Linux 主机上面，而这四组人都是你管的，但是这四组人又互相没有信息的交流，那么你就可以进行这四组人的邮件群组功能，同时，将你的实体账号分别加入这四个群组中！哈哈！就可以收到这四个群组的信件啰！

二、使用者的别名设定与重要邮件备份：

除了群组功能之外， aliases 还可以用来做为一个用户多个邮件名称的设定喔！例如，小老弟我，鸟哥的浑号仅在 Linux 里面通称而已，一般的上班单位里面，仍然主要以我的名字记忆我的邮件的！也就是说，我具有两个账号在我的 tsai.adsldns.org 上面，分别是 vbird@tsai.adsldns.org 及 dmtsai@tsai.adsldns.org ！那么我是否还要建造另一个实体使用者账号呢？当然不需要，我只要在 /etc/mail/aliases 里面加入一行：

```
1. 编辑别名设定档：
[root@test root]# vi /etc/mail/aliases
dmtsai:    vbird

2. 制作数据库 /etc/mail/aliases.db
[root@test root]# newaliases
```

那么未来这两个电子邮件均将寄到我的 /var/spool/mail/vbird 信箱里面去喔!所以,不论是寄给 dmtsai 还是 vbird ，我都可以直接以 vbird 这个账号来取得这两个邮件地址的信件，因为都这两个邮件都放到 /var/spool/mail/vbird 这个信箱嘛！相当的方便吧！ ^_^另外，如果我要将某个账号在收信时，顺便备份一份到系统当中时，例如寄信到 vbird 时，顺道寄一份到 testing 时，可以这样做：

```
1. 编辑别名设定档：
[root@test root]# vi /etc/mail/aliases
dmtsai:    vbird
vbird:     vbird,testing

2. 制作数据库 /etc/mail/aliases.db
[root@test root]# newaliases
```

如此一来，则寄给 vbird 的信件，vbird 自己保留一份之外，也会再寄给 testing 这个人喔！可以做为备份的需要啦！

三、外部信件的寄送

另外，如果你的电子邮件想要将该邮件外传的话，要怎么做？这个同样可以做为邮件的『异地备援』之用！怎么说呢？同样用我们的 tsai.adsldns.org 来做说明吧！假设我的账号 vbird@tsai.adsldns.org 送到 tsai.adsldns.org 之后，要再传送一份给 vbird@vbird.adsldns.org ，也就是说，信件传送到 tsai.adsldns.org 这部主机的 vbird 后， tsai.adsldns.org 会主动的再将该信件外传到 vbird.adsldns.org 这一部上面去！怎么做呢？你可以这样搞定：

```
1. 编辑别名设定档：
[root @test root]# vi /etc/mail/aliases
dmtsai:    vbird
vbird:     vbird,vbird@vbird.adsldns.org

2. 制作数据库 /etc/mail/aliases.db
[root @test root]# newaliases
```

如此一来，任何人寄给 vbird@tsai.adsldns.org 的邮件，都会额外再多寄一份给 vbird@vbird.adsldns.org !就可以达到异地备援的目的啦!很方便吧!此外,你也可以用来做为 Mail list 呢！嘻嘻！

四、档案类型的别名 include

再让我们回到第一点『群组寄信』的地方，您会不会觉得，如此一般的设定方法，在经过了几年之后，你的这个 aliases 会变的乱七八糟的！所以这里再让我们学个有用的技巧，就是利用 aliases 里面的 include（包括）功能，使用档案类型的方法来达成群组寄信的目的！举上面学校相同的例子来说明好了，今天我的 student9213 这个群组账号中，所有的人员都给他写入 /etc/mail/student9213 这个档案当中，然后再以 include 的功能给他写入 aliases 这个档案中～你可以这样做：

```
1. 编辑 /etc/mail/student9213 ：
[root@test root]# vi /etc/mail/student9213
st001, \
st002, \
st003, \
st004, \
....
st050
假设共 50 个学生，则最后一个不用加上『 , \』！与变量设定规则相符！

2. 还是要编辑 aliases 的！
[root@test root]# vi /etc/mail/aliases
dmtsai:        vbird
vbird:         vbird,vbird@vbird.adsldns.org
student9213:   :include:/etc/mail/student9213

3. 制作数据库 /etc/mail/aliases.db
[root@test root]# newaliases
```

整个写法是：
群组账号： ：include：使用的档案完整档名
请注意，这个档案类型的格式为『:include:』亦即 include 两边都有冒号，并且，在最前面账号的地方也有冒号喔！不要记错了～至于在 /etc/mail/student9213 这个档案中的写法与 aliases 后面接的账号或 E-mail 类型写法相同，例如：
vbird, vbird@vbird.adsldns.org, userID@host.domain.name
不过，我们也可以利用跳脱字符『\』来加以格式美观化，会比较整齐画一喔！但是，仍然不要忘记了那个可爱的『逗号』喔！例如上面表格里面的 st001, ... 说明的样式！这部份如果不太明了的话，请拿出鸟哥的私房菜 Linux 基础学习篇，好好的看一看 BASH Shell 里面介绍的变量设定规则吧！ ^_^！

上面提到的都是关于系统管理员设定的数据部分，那么预设的 aliases 里面有什么东西呢？通常有这些数据存在的喔：

```
[root@test root]# vi /etc/mail/aliases
# 基础 sendmail 数据！由于 sendmail 预设使用 mailer-daemon 与
# postmaster 做为数据发送者，或者是信件被退回时的账号！但是我
# 们的系统并没有这两个账号，因此，必需要使用 aliases 的功能！
# 如果是使用 sendmail ，那么底下这两行『务必存在』才行！
```

```
mailer-daemon:  postmaster
postmaster:     root

# pseudo accounts. 也就是系统的账号，这些账号是给系统来使用的，
# 基本上，这些账号并无法登入主机，但是偏偏某些程序进行时，产生
# 的错误讯息可能会寄给该系统账号，但该账号无法登入，所以会让系统
# 无形之中遗失许多的信息，所以啰，这些账号也需要来做 aliases
# 并且将收件者交给系统一定会有的人物！ root 是耶！通常这些账号
# 常见的有 bin, daemon, adm, lp, sync, shutdown, halt, mail, news
# uucp, operator, games, gopher, ftp, nobody, named, xfs, system,
# 等等等等！
bin:            root
daemon:         root
adm:            root
lp:             root
sync:           root
shutdown:       root
....(略).....

# trap decode to catch security attacks 有些攻击者在攻击你的主机时，
# 该相关的信息会寄给你的 decode 这个账号，将他转成 root 吧！
decode:         root

# 这是 root 的收件信者！ 由于预设状况中， root 是不能在主机外部
# 的任何一部计算机收信的！如果您想要让你的一般账号可以接收 root 的
# 信件，以实时掌握主机信息，那么底下的 # 将他打开，后面接你的
# 账号吧！
#root:          your_account
```

这些资料在 aliases 当中是必需要的喔！如果你是自行以 Tarball 建立 Sendmail 的话，那么这个 aliases 可是需要加入的呐！

- 什么是 Mailling list ：我们刚刚在 aliases 里面有进行过群组寄信对吧！那么你寄给某个账号时，该账号会将你的来信再寄给该群组账号的所有人员，此外，还可能将该封信件也备份一份在自己的机器上，这个功能就可以称为是 Mailling list 啦！有点像是目前很流行的『电子报』之类的咚咚！也就是说，我们可以用这个很简单的 aliases 这个档案就可以达到 Mailling list 的功能了！

设定邮件转递 ~/.forward

了解了 aliases 之后，是否会发现一个问题呢？那就是，虽然 aliases 可以帮我们达到 mailling list 的功能，但是『只有 root 才可以修改该 aliases 档案』，那么万一我并不是网站管理员，怎么办？是否还是可以建立一个 mail 转寄的功能呢？确实还是可以啦！这个时候可以使用邮件转寄 (mail forward)

的功能喔！你可以在该账号的家目录之下建立一个档案，档名为 ~/.forward ，利用该档案就可以达到 Mailling list 的功能啦！

还是来假设个案例啰：假设我有一个账号，名称为 birdhouse ，而我希望寄信给该 birdhouse 时，就可以将信件分送给该 MTA 主机上面的 bird1, bird2, bird3, 及 bird4 之外，还可以寄给外部的 bird@yahoo.com 及 bird@pchome.com 此外，还记录一份给 birdhouse 这个主要账号！这个时候你可以这样做：

```
[birdhouse@test birdhouse]$ cd ~
[birdhouse@test birdhouse]$ vi .forward
birdhouse
bird1
bird2
bird3
bird4
bird@yahoo.com
bird@pchome.com
[birdhouse@test birdhouse]$ chmod 644 .forward
```

直接将你要寄出去的邮件地址都写到 ~/.forward 里面去，每个地址都占用一行，如此一来，嘿嘿！只要是寄给 birdhouse@tsai.adsldns.org 的邮件，就可以自动的来传送到 ~/.forward 内部所设定的邮件地址啰！同时，这个档案除了可以用来建立类似 mailling list 的功能外，也可以让你自己设定『异地备份邮件』的功能呢！就是在该档案内写入你自己的账号以及外部信件的邮件地址，那么当你的主机收到要给你的信时，除了给你一份外，还会再转一份给你订定的邮件地址！而最大的优点是，『不需要建立数据库或者重新启动 sendmail ！』以 vi 设定完，并且储存后，立刻生效！好用的很～

不过需要注意的是，由于这个档案是这样的方便设定，万一被某些居心不良的人看到甚至可以修改时，那可就不得了了！你能想象你的信件都会被复制一份到某人的信箱吗？所以啰，这个档案必需只有你能修改，其它人则仅能查阅而无法修改才行喔！亦即这个 .forward 档案的权限必需要：

1. 该档案所在使用者家目录权限，其 group、other 不可以有写入权限。
2. .forward 档案权限，其 group、other 不可以有写入权限。

察看信件队列 (mailq)与 Mailers 状态

关于信件队列：
对于 Sendmail 设定到目前为止，应该也可以正常的应付蛮多工作的啦！但是我们还是得要了解一下的是：『如果我将邮件送到 Sendmail 主机后，Sendmail 便会帮我将该封邮件传送到目的地的 MTA ，不过，如果目的地 MTA 主机有问题时，这封信会怎么跑？』一般来说，如果 DNS 设定正确的话，也就是说目的地 MTA 有 MX 标志存在时，只是刚好这部主机暂时无法联机，或者是有些问题，导致无法立即接受来自你的 MTA 的邮件，此时这封邮件将会被放置到你的 MTA 主机的队列目录去，通常预设是在 /var/spool/mqueue 当中！然后在一定的周期时间内， Sendmail 会定时的尝试将邮件寄送出去，一般 Sendmail 的预设设定是：

1. 如果该封信在五分钟之内无法寄出，则系统会发出一封『警告信』给原发信者，告知该封邮件尚无法被寄送出去，不过，系统仍会持续的尝试寄出该封邮件；
2. 如果在四小时候仍无法寄出，系统会再次的发出警告信给原发信者；
3. 如果持续进行五天都无法将信件送出，那么该封邮件就会退回给原发信者了！

在 Red Hat 的预设条件中，在 /var/spool/mqueue 当中的信件会每隔 60 分钟由 Sendmail 尝试重新传送一次到目的地去！这个尝试的时间是可以改变的！可以利用 sendmail 的指令或者直接修改 /etc/sysconfig/sendmail 里面的 『QUEUE=时间』来修订！例如，如果你想要让 Sendmail 每隔 30 分钟就帮你尝试传送 /var/spool/mqueue 里面的未寄出的信件时，那么就将 /etc/sysconfig/sendmail 这个档案里面的『QUEUE=1h 』改成『QUEUE=30m 』即可！

信件队列的内容：
老实说，信件队列的内容是给 Sendmail 看的，不是给人看的，所以我们都不可能看的懂他的讯息！这个时候，只得以 Sendmail 的指令来反查这些邮件队列到底是什么咚咚了！很简单的，只要下达 mailq 或者是 sendmail -bp 就可以这些邮件队列的基本数据！

```
[root@test root]# mailq
                /var/spool/mqueue (1 requests)
----Q-ID---- --Size-- -----Q-Time----- ------------Sender/Recipient------------
h1LEKYR23711    36414 Fri Feb 21 22:20 <gold@tsai.adsldns.org>
                 (Deferred: Connection refused by vbird.adsldns.org)
                                       <qqq@vbird.adsldns.org>
```

Q-ID：表示此封邮件队列的代表号 (ID)；
Size ：这封信有多大容量 (bytes)的意思；
Q-Time：这封信什么时候进入 /var/spool/mqueue 这个目录的，并且说明无法立即传送出去的原因 (例如上面的 Deferred)；
Sender/Recipient：送信与收信者的电子邮件啰！

如果您有开放邮件的话，那么记得偶而要去看一看您的邮件队列 (mailq) 是否存在大量的未寄出信件喔！好让你知道是否可能被当作转信站啦！

关于邮件在 Mailer 中的统计状态 (mailstats)
除了 mailq 记录了在信件队列的信息之外，还有一个档案可以纪录 sendmail 由『开始运作到目前为止，邮件的收发总计资料』喔！预设就是 /etc/mail/statistics 这个属性为 data 的档案，那么我怎么将这个档案的数据读出来呢？很简单啊！就藉由 mailstats 这个小指令来读取即可！读取出来的结果有点像这样：

```
[root@test root]# mailstats
Statistics from Sat Mar 23 21:34:09 2002
 M   msgsfr  bytes_from   msgsto    bytes_to  msgsrej msgsdis  Mailer
 4    50752    9126380K    23617    5425714K     1070     122  esmtp
 9    21329    5919236K    65162   13364494K     1068       8  local
```

```
========================================================
 T     72081    15045616K     88779    18790208K       2138       130
 C     72081                  88779                    2655
```

上面共出现七行，第一行只是显示目前的时间而已，至于每个直列的意义为：

- M　　：只是一些邮件工作(Mailer)代号的标题啦！不过，重要的地方在第六行的 T ，那个是『Total 总和』的意思～
- msgsfr:共有多少封信由这个邮件工作 (mailer) 所发出去的呢？以第四行最右边看到的 local 为例，这表示由 local 发出的信件共有 21329 封的意思～
- bytes_from：表示的是信件数据容量，同样以第四行 local 为例，他发出的 21329 封信，共有 5919236K 喔！
- megsto：与 msgsfr 类似，只是 msgsfr 是寄出数据，而 megsto 则是『收到的信件封数』以上面的数据来看，则 local 收到的共有 65162 封信！
- bytes_to：这就不需要解释了吧！ ^_^
- msgsrej：那个 rej 是 reject (拒绝) 的意思，这一列是信件被 deny 的次数；
- msgsdis：那个 dis 是 discard 的意思，同样是 deny ，只是经由 discard 的程序就是了！
- Mailer ：就是 sendmail 许多 mailers 当中的一个啦！那个 esmtp 主要用来对外，至于 local 则主要针对本机端的 mailbox 啰！

由上面的资料你会发现，哇！怎么鸟哥的信箱 15GB 的信件啊！真可怕～别担心，那个数据是由『 sendmail 开始运作到现今』的结果，我的 sendmail 运作了若干年了，有这样的信件资料量其实不怎么吃惊啦！ ^_^

客户端的使用说明

设定 Mail Server 就是要拿来用的！所以，当然要介绍一下怎么使用 Mail Server 啦！我们分为 Linux 与 Windows 稍微做介绍啰！

- Linux 下使用 mail 功能

  在 Linux 的系统当中，一定会存在的客户端当中的邮件指令就是 mail 这个指令啦！由于这个 mail 指令是直接使用 sendmail 的 local 端的功能，所以使用 mail 时，即使你的 port 25 没有启动，仍然可以寄信喔！(但是在 Red Hat 9 以后，已经将这样的功能取消了！所以您至少需要启动 sendmail 的预设设定，亦即仅监听 127.0.0.1 的 port 25 才行！)早期没有 POP3 这个协议的服务时，用户在使用 mail 都需要登入主机之后，才能以 mail 这个指令来操作邮件呢！现在就幸福多了，可以直接用 Netscape 之类的软件来收发电子邮件说。来谈一谈怎么发信与收信吧！

  - 用 mail 直接编辑文字邮件与寄信：

    使用 mail 最简单的方式就是直接的使用在线编辑的方式来将文字数据传送出去啰！假如你要发一封信给 vbird@qqdomain.name 时，你可以使用『 mail user@email.domain.name 』的格式，所以你可以这样做：

```
[root@test root]# mail vbird@qqdomain.name
Subject: This is a test mail
```

```
There are writing area!
You can't use the Up/Down button in this form...
you can finish with "."
.  <==这个『.』就是结束符号！要正常离开编辑画面就是 . ！
Cc: <==这就是副本！
```

- 如同上面的样式，mail 会主动的显示 Subject 给你，你可以输入这封信的标题在此～然后就进入编辑画面啦！你可以在编辑画面中写入中英文喔！之后，最重要的是，一定要在开头的地方输入那个句号『.』，这样 mail 就会开始将信传送出去。在 Red Hat 的预设状况中，还包含提供了一个副本收受者的邮件地址，就是出现 Cc: 那边，你可以在后面再接上另一个邮件地址，那就是副本啦！这样就将信寄出啰！

- 使用 IP 测试寄信：
万一你的 Linux server 并没有 domain name 时，是否就无法建置 Mail Server 了呢？当然不是啦！还是可以架设 Mail Server 的啦，还记得我们在前面的网络基础里面有提过啦，既然有内部 lo 的接口，自然就一定可以让我们『测试 tcp/ip 架构』的设定啦！既然如此，我又没有 domain name ，又架设了一个提供只有少数人知道的 mail server 时，那么人家怎么寄信给我啊！呵呵！很简单啊！就利用 IP 来寄信啊！怎么寄出去？

```
[root@test root]# mail vbird@[127.0.0.1]
[root@test root]# mail vbird@[192.168.0.100]
```

- 看到了吗？就是将 domain name (FQDN) 以 IP 来置换，而且，使用中括号将 IP 包起来，这个中括号很重要啊！不要忘记了喔！使用 IP 寄信仅是一个权宜之计，毕竟没有多少人会记得你的主机 IP 啊!并且,若是使用 ADSL 计时制的拨接方式,那么得到的 IP 是 ISP 动态分配的 (dynamic)，所以可能每次上网的 IP 都不固定～那么不是会搞死人....所以啦，申请一个 domain name ，如果每次上线都会不同 IP 的情况下，也可以申请免费的动态 DNS 系统的领域名称,而使用国内的 ISP 提供的主机名称自动对应 IP 的功能也可以啦！这些名称都可以用来架设 mail server 喔！

- 用 mail 寄出纯文本文件：
还记得鸟哥的 Linux 私房菜 -- 基础学习篇里面的 Bash shell 提到的标准输入 (<) 吗？既然我使用 mail 的时候需要用到键盘输入，那么我当然可以利用『标准输入』来使档案替代键盘的 Keyin 啰!因为在 mail 的编辑画面中,我们无法使用上下左右按键,而且，如果你刚刚在上一行写错字了，还没有办法回到上一行呢！编辑上面实在不怎么人性化。所以，我可以使用任何的文字编辑软件，将我的信件编辑成档案，请注意，必需是纯文字文件喔!然后再将该档案寄出即可!例如我要将 root 家目录底下的 .bashrc 寄出去给 vbird ，可以这样做：

```
语法：
[root@test root]# mail -s  '这里可以接邮件标题'  这里是邮件收件者  <  文件名称
范例：
[root@test root]# mail -s 'This is a test mail' vbird < /root/.bashrc
```

- 需要注意的是，如果 mail 给账号而已，那么就是寄给本机的使用者，如果是以电子邮件地址的写法，才是向外面寄出去的邮件喔！

- 用 mail 接收 mailbox 的信件：
  寄信还比较简单，那么收信呢？同样的啦，收信还是使用 mail ，直接在提示字符之后输入 mail 时，会主动的捉取使用者在 /var/spool/mail 底下的邮件信箱（mailbox），例如我 vbird 这个账号在 shell 的环境中，输入 mail 后，就会将 /var/spool/mail/vbird 这个档案的内容读出来，并且显示给 vbird 看！

```
语法：
[vbird@test vbird]# mail
范例：
[vbird@test vbird]# mail
Mail version 8.1.1 6/6/93.  Type ? for help.
"/var/spool/mail/vbird": 2 messages 2 new
>N  1 root@vbird.adsldns.o  Sat Feb 22 13:01  24/945   "test uuencode"
 N  2 root@vbird.adsldns.o  Sat Feb 22 13:12  26/838   "This is a test mail"
&
```

- 在上面的画面中，显示 vbird 共有两封『新信』此外，底下会附上这两封新信的发信站与标题及时间等。

  - 读信：有看到『>』那个符号吧！那表示目前 mail 所在的邮件字段，你可以直接输入 Enter 即可看到该封信件的内容！另外，你也可以在『&』之后的光标位置输入号码，就可以看该封信件的内容了！(注：如果持续按 Enter ，则会自『 > 』符号所在的邮件逐次向后读取每封信件内容！)
  - 显示标题：如果要重新显示每封信的标题，可以输入 h 即可；
  - 回复邮件：如果要回复目前『 > 』符号所在的邮件，直接按下『R』即可进入刚刚前面介绍过的 mail 文字编辑画面啰！你可以编辑信件后传回去啰！
  - 删除邮件：按下『 d## 』即可删除邮件！例如我要删除掉第 2 封邮件，可以输入『d2』如果是要删除第 10-50 封邮件，可以输入『d10-50』来删除喔！请记得，如果有删除邮件的话，离开 mail box 时，要使用『q』才行！
  - 储存邮件到档案：如果要将邮件资料存下来，可以输入『 s ## filename』，例如我要将上面第一封邮件存下来，可以输入『 s 1 uuencode 』即可将第一封邮件内容存成 uuencode 这个档案！
  - 离开 mail ：要离开 mail 可以输入 q 或者是 x ，请注意『输入 x 可以在不更动 mail box 的情况下离开 mail 程序，不管你刚刚有没有使用 d 删除数据；使用 q 才会将删除的数据移除，并且会将所有已读过的信件内容转存到你家目录下的 mbox 档案！』也就是说，如果你不想更动 mail box 那就使用 x 或 exit 离开，如果想要使刚刚移除的动作生效，就要使用 q 啦！不过，使用 q 之后，只有未读的信件才会保留在 /var/spool/mail/accout 里面，其它已读的数据都会被存入 ~/mbox 当中！例如 /home/vbird/mbox 为储存 vbird 已读过的信件！

- 请求协助：关于 mail 更详细的用法可以输入 help 就可以显现目前的 mail 所有功能！

上面是简易的 mail 收信功能！不过，如果离开 mail 时按下 q ，不是会有信件转存到 ~/mbox 这个邮件信箱吗？那么我要如何读取这个档案内的信件数据呢？可以简单的使用这个方式来读取：

```
[vbird@test vbird]# mail -f ~/mbox
```

使用『-f file 』规定新的 mail box 档案，如果没有 -f file 的话，就会直接使用 /var/spool/mail 里面的 mailbox 啦！

- 设定检查邮件的间隔时间 MAILCHECK：
  如果在你的 Shell 环境下有新信进来时，通常我们的 bash 会很好心的告诉你有来信了！预设的条件中是 60 秒钟检查一次 /var/spool/mail 当中是否有新信！如果想要改成 30 秒呢？可以的！直接设定『 MAILCHECK 』(大写字母)这个变量即可！

```
[vbird@test vbird]# MAILCHECK=30
```

- 或者直接将该变量写入你的 ~/.bashrc 当中亦可！

- 用 mail 夹带档案寄信与收信：
  上面的方法都仅提供『纯文字』的寄件而已，如果我想要『夹带档案』在邮件当中呢？可以使用 mail 这个指令来达成吗？呼呼！确实是可行的！不过，单纯的 mail 无法达到这个目的，我们还必需要使用档案译码 （ encode ） 的功能才能达到这个夹带档案的目的喔！不过，要使用 uuencode 的功能，就得要安装 sharutils 这个套件！如何安装呐？！很简单啊！拿出光盘，直接安装，或者使用 APT 或 YUM 来安装！都可以啦！例如：
  apt-get install sharutils
  yum install sharutils

```
寄信：
[vbird@test vbird]# uuencode 欲夹带的档案文件名 编码的标题 | mail -s 'title' 收件者
范例：
[vbird@test vbird]# uuencode ~/.bashrc bashrc | mail -s 'test uuencode' \
> vbird@tsai.adsldns.org
```

- 这样信件就送出去啦！而且是附件夹带喔！你可以使用 Netscape 或者 Outlook 之类的软件直接收受该档案，如果是在 shell 环境中，你可以这样做：

```
1. 储存该邮件使成为档案：
[vbird@test vbird]# mail
Mail version 8.1.1 6/6/93.  Type ? for help.
"/var/spool/mail/vbird": 2 messages 2 new
>N  1 root@vbird.adsldns.o  Sat Feb 22 13:01  24/945   "test uuencode"
& s 1 testfile
目前的目录下会产生一个 testfile 的新档案

2. 将捉下来的档案解碼
[vbird@test vbird]# uudecode testfile -o outfile
                    uudecode 读入档案    输出的档案
```

- o 如此一来， outfile 就是解碼后的档案内容啦！至于 testfile 内容有点像这样：

```
From root@vbird.adsldns.org  Sat Feb 22 13:01:05 2003
Return-Path: <root@vbird.adsldns.org>
Received: from vbird.adsldns.org (localhost [127.0.0.1])
        by vbird.adsldns.org (8.12.7/8.12.7) with ESMTP id h1M514Jc022610
        for <vbird@vbird.adsldns.org>; Sat, 22 Feb 2003 13:01:05 +0800
Received: (from root@localhost)
        by vbird.adsldns.org (8.12.7/8.12.7/Submit) id h1M514dg022608
        for vbird; Sat, 22 Feb 2003 13:01:04 +0800
Date: Sat, 22 Feb 2003 13:01:04 +0800
From: root <root@vbird.adsldns.org>
Message-Id: <200302220501.h1M514dg022608@vbird.adsldns.org>
To: vbird@vbird.adsldns.org
Subject: test uuencode
Status: R

begin 666 bashrc
M(R`N8F%S:')C"@HC(%-O=7)C92!G;&]B86P@9&5F:6YI=&EO;G,*:68@6R`M
M9B`O971C+V)A<VAR8R!=.R!T:&5N"@DN(")E=&,O8F%S:')C"F9I"@I0051(
M/2]S8FEN.B]U<W(O<V)I;CHO8FEN.B]U<W(O8FEN.B]U<W(O6#$Q4C8O8FEN
M.B]U<W(O;&]C86PO8FEN.B]U<W(O;&]C86PO<V)I;CHO=7-R+VQO8V%L+V%P
M86-H93(O8FEN"F5X<&]R="!0051(("@IA;&EA<R!L;3TG;',@+6%L?&UO<F4G
""@H`
`

end
```

- o 真正的档案内容是 begin 到 end 的那些粗体字！使用 uudecode 时， uudecode 会主动的分析 begin 开始的字段，然后将 begin 后面的内容解译还原成为原来的档案内容！呵呵！恭喜您！

-
-

- Linux 下使用 telnet 功能

  我们上面提到的是 mail 使用 sendmail 的预设功能，那是属于 local mailer 的功用！那么我是否可以藉由 smtp 来寄信呢？亦即使用 SMTP 这个服务来寄信！可以的啦，那就必需要使用 telnet 的功能啰！

```
1. 寄信：
[vbird@test vbird]# telnet localhost 25
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
220 vbird.adsldns.org ESMTP Sendmail 8.12.7/8.12.7; Sat, 22 Feb 2003 13:52:52 +0800
ehlo localhost <==必需先跟主机打招呼喔！
250-vbird.adsldns.org Hello localhost [127.0.0.1], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-DELIVERBY
250 HELP
mail from: "Myname"<vbird@vbird.adsldns.org>
这里输入我要寄信的来源邮件地址，请记住，前面""里头是显示在信件表头的说明，
后面< >里面接的才是你真正的邮件地址喔！
250 2.1.0 vbird@vbird.adsldns.org... Sender ok
rcpt to: <root@vbird.adsldns.org> <==输入邮件送达的目的端邮件地址
250 2.1.5 root@vbird.adsldns.org... Recipient ok
data  <==要开始写信了喔！告知 data ！
354 Enter mail, end with "." on a line by itself
This is just testing
.              <==这两行是内容！不要忘记 . 这个咚咚！
250 2.0.0 h1M5qqJc022990 Message accepted for delivery
quit<==离开 telnet 程序！
这样信件就寄出去啦！
```

- 这样就 OK 啦！ ^_^！

  注：Linux 下使用 telnet 功能，在 MAIL FROM、RCPT TO 内，后面接的项目若是 e-mail 的话，应该使用 "< >" 包起来，这样子才正确。

  - MAIL FROM: usersname<userid@hostname.domainname>

    MAIL FROM: "users name"<userid@hostname.domainname>

- 另外许多 MTA 目前都会限制再使用 MAIL FROM 时要先使用 HELO 或者是 EHLO 先打招呼过，要不然会不允许进行后续沟通。

-
- X-Window 与 Windows 的 MUA 功能

Outlook 与 Netscape 会用吧？！由于是全图形接口，所以也没有什么好特别说明的地方，底下就来说一说关于 outlook 的设定方法，其它的 MUA 接口设定的方法都差不多啦！仔细参考一下即可啰！

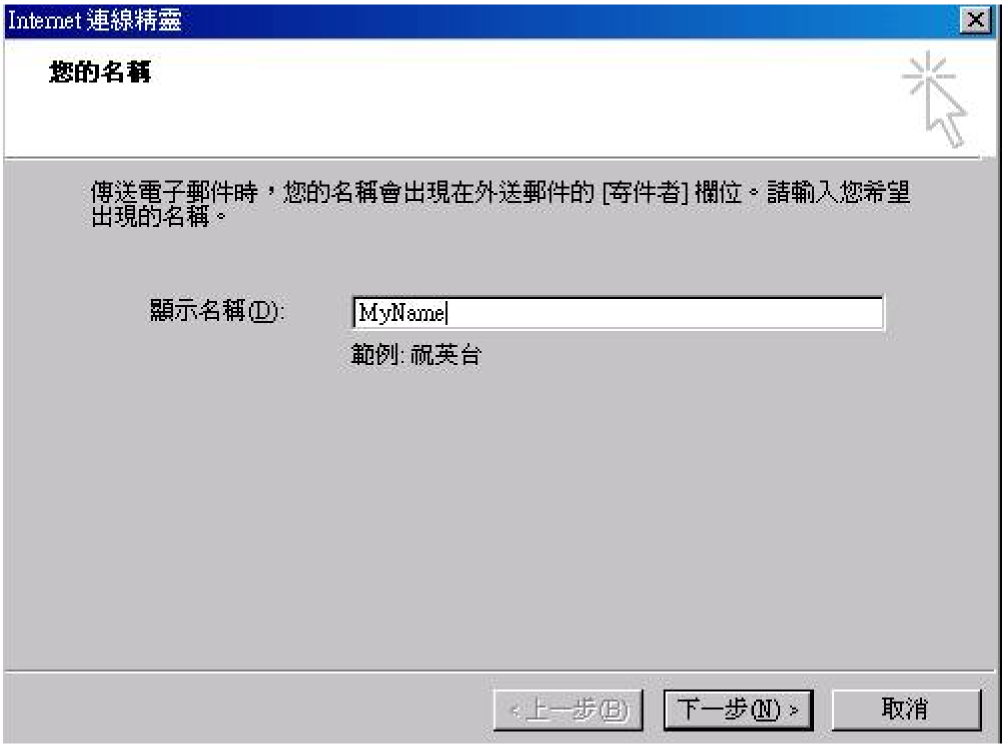
1. 开启 OutLook Express，点选『工具』里面的『账号』；



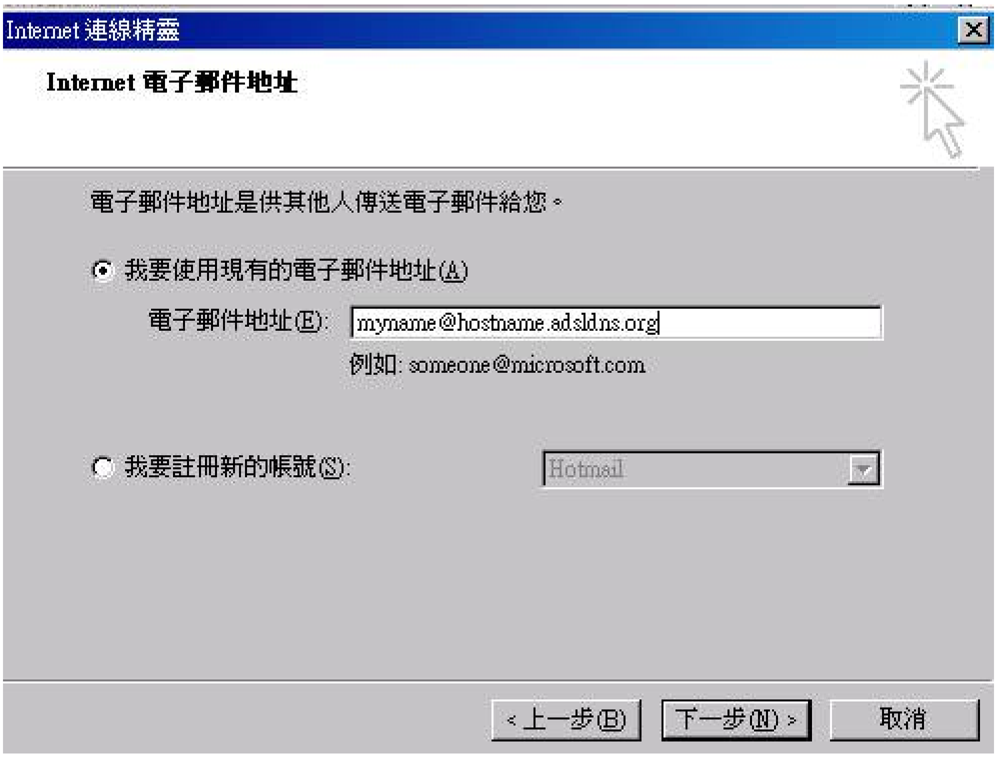
2. 点选『新增』选择『邮件』来设定；



3. 输入显示名称：这个名称是别人在收信的时候，可以看到的寄件人称谓；



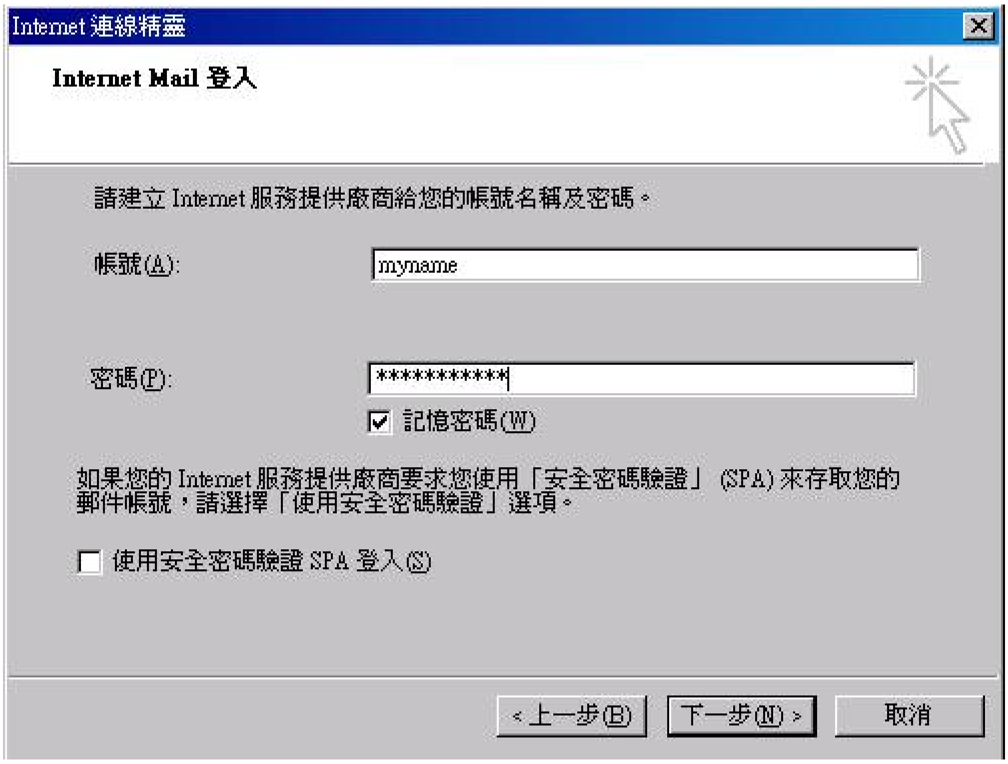
4. 接着下来比较重要了，请使用你的 ID@hostname.adsldns.org 登入你的邮件服务器；



5. 然后，当然邮件服务器收发都是同一个，请使用你的 DNS 申请的名称；



6. 输入账号与密码，也可以选择『记忆密码』，然后按下一步，就完成了。



- 基本上，Windows 这边只要设定如此即可！然后你就可以用你的 Linux 来当邮件服务器啰！！

关于邮件主机安全的设定

Sendmail 常常被传成『安全性很差的邮件服务器！』当然， Sendmail 官方网站也很不满的响应说『其实，Sendmail 的问题来自于一些"该死的"档案权限设定错误的状况！』也就是说，其实 sendmail 常常会因为『系统管理员』设定档案或者目录不良的情况下，导致 Mail Server 发生问题啰！所以，底下我们就来说一说这个 sendmail 的安全性与其它相关的安全方面设定吧！

---

sendmail 本身的安全设定项目(Sendmail 官方的建议)

sendmail 本身已经提供相当多的安全项目建议，其中，大部分是在于『目录与档案权限』的设定要求上面！

- 请确定 aliases 这个档案的权限，仅能系统信任的 User 可以存取，通常其权限为 644 ；
- 请确定 sendmail 读取的数据库 (多半在 /etc/mail 底下的 *.db 档案)，例如 mailertable, access, virtusertable 等等，仅能由系统信任的使用者读取，其它一概不能读取，通常权限为 640 ；
- 系统的队列目录 (/var/spool/mqueue) 仅允许系统读取，通常权限为 700 ；
- 请确定 .forward 这个档案的权限也不能设定成为任何人均可查阅的权限，否则您的 e-mail 数据可能会被窃取～
- 总之，一般用户能够不用 .forward 与 aliases 的功能，就不要使用！

不过，这样的设定自然不够安全的啦！在 Sendmail 8.12 版本以后，甚至严格的规定系统的操作者为 smmsp 这个使用者呢！会更加的安全一些～如果您有兴趣的话，不妨将您的 sendmail 以底下介绍的 Tarball 的方式升级更新到 sendmail 8.12 版喔！

SMTP 认证

由前面的 Sendmail 的设定说明当中，您是否觉得上面所设定的 Sendmail 已经够用了呢？呵呵！想必是不够的！鸟哥我就觉得，嗯！还是有待加强！怎么说呢？想个例子好了，由于我的 sendmail 为了 Open Relay 的问题，所以势必要关闭所有的 relay ，而仅开放 Mail 本机以及部分网域或者主机的 Relay 使用权！不过，我总有在外面工作的时候吧！我总是有使用计时制 ADSL 这种非固定 IP 的联机模式来上网的时候吧！我的 Mail Server 上面的用户总有使用 notebook 而到处跑得吧！如此一来，这个 /etc/mail/access 的设定势必是不够的～那要怎么办呢？

想象一下，如果我的 sendmail 也跟 pop3 这个收信服务一样，在要求传信的时候，必须提供账号与密码让 Mail 主机来认证，如此一来，不就可以藉由这账号、密码的机制来限制使用者的使用了吗？！呵呵！好样的！没错，确实可以使用这样的机制。事实上，目前较常被使用的这种认证机制为 SMTP 邮件认证机制 (SMTP Authorization) ，主要透过 cyrus sasl 这个套件来达成邮件的认证动作，那么到底他是如何达到认证的呢？

- Cyrus SASL 的动作：

  Cyrus Simple Authentication and Security Layer 简称为 Cyrus-SASL ，他最大的功能在提供一些应用程序所使用的认证函式库！这里头最有名的例如 Sendmail 这个邮件服务器了！ 应用程序可以透过 SASL 所提供的函式库功能，并且定义出认证的方式，让 SASL 透过与主机的沟通动作，提供应用程序来达到认证的目的！举个例子来说，如果我的 sendmail 已经提供了 SMTP 认证的功能，那当使用者进入认证阶段时：

  1. 首先， sendmail 会去取用 SASL 的函式库资料；
  2. 此外，由于 SASL 可以进行的认证机制相当的多，所以 sendmail 必须要指定 SASL 的认证方式，一般而言，我们都会直接以 /etc/shadow 里面的账号密码来进行认证！至于针对 sendmail 的 SASL 认证方法则预设设定在 /usr/lib/sasl/Sendmail.conf 或 /usr/lib/sasl2/Sendmail.conf (根据 cyrus SASL 版本的不同而异！)
  3. SASL 根据设定的方法去取用密码与账号内容,并且加以比对,响应给 sendmail 该次比对是否成功！

  基础的流程是这样，不过 SASL 除了 Sendmail 的认证模式之外，其实他还提供很多的功能啦！因为 SASL 主要就是一个函式库，而这个函式库还具有额外的提供『密码认证档案』的功能，所以，只要是能够支持 SASL 的应用程序，就可以利用 SASL 所提供的这个认证功能来达到共享同一认证的好处！例如同一个 user 在 LDAP 与 Sendmail 这两个软件中，均可以使用同一组账号与密码！而不需要额外的针对不同的套件来设定额外的账号与密码！

  若单纯的指 Sendmail 这个应用程序的话，那么 SASL 至少可以提供使用 /etc/shadow 与 PAM 这两个密码验证机制，不过请特别留意的是， cyrus SASL 目前已经出到第二版了，而现今的 sendmail 大部分还是以 1.5.xx 版本进行设定的。实际上，这两个版本的认证原理与方法虽然一样，但是认证使用的执行档已经不同了！所以在 /usr/lib/sasl2/Sendmail.conf 这个档案当中的设定内容也不一样了！如果您要设定 SMTP 的 AUTH 的话，请特别留意这个不同点喔！

- 实作的流程说明：

  由前面的介绍我们知道，要使用 SMTP AUTH 这个功能，你必须要：

  1. 你的 Linux Server 必须要已经安装 Cyrus SASL 函式库；
  2. 你的 /usr/sbin/sendmail 这支程序必须要将 SASL 的函式库功能编译在内；
  3. sendmail 设定档 (sendmail.cf) 必须要将 SASL 的功能启动 (注: /usr/sbin/sendmail 虽然已经将 SASL 函式库编译在内，但是 sendmail.cf 这个档案仍然要将 SASL 的支持启动之后，才能使用 SMTP)；
  4. 必须指定 sendmail 的 SASL 认证模式，通常有 pam 与 shadow 两种模式，在 SASL version 1 当中， sendmail 的 SASL 认证模式设定文件为 /usr/lib/sasl/Sendmail.conf (S 为大写)，若为 SASL version 2 则在

/usr/lib/sasl2/Sendmail.conf ！而如果 Sendmail 里面设定为 pam 的话，那么 /etc/pam.d 里面亦需存在 smtp 这个档案才行！(所以我们通常都使用 shadow 而已～)

5. 重新启动 sendmail 即可！

可以看得出来，要让你的 Sendmail 由无到有的支持 SMTP AUTH 的话，那么就必须要重新编译 /usr/sbin/sendmail 这个执行档！哇！那就必须要使用 Tarball 的方式，并且依照上面的动作，一步一步的进行啦！过程还颇为繁复呢！这个动作我们在底下 以 Tarball 完整安装具有 SMTP 的 sendmail 部分再来讲解，如果您的 sendmail 并非为 Red Hat 的话，就得前往 Tarball 安装 sendmail 的章节了，不过，如果是 Red Hat 呢？恭喜您啦！因为 Red Hat 的 Sendmail 预设已经包含了 SASL 的函式库功能，只是预设的参数设定档 (sendmail.cf) 并没有启用这个功能而已～底下我们就先以简单的 Red Hat 的 sendmail 来说明启动 SMTP AUTH 的流程吧！

- Red Hat 的 SMTP 认证启用流程：
  很棒的是， Red Hat 的 sendmail (/usr/sbin/sendmail 执行档) 已经支持 Cyrus SASL 了，只要将 sendmail.cf 里面关于 SMTP 认证的功能启动即可！所以，只要去编辑 /etc/mail/sendmail.mc 即可 (注：这个档案在不同的版本中，也会有被放在 /usr/share/sendmail-cf/cf/redhat.mc 的时候！反正只要找到 sendmail.mc 或 redhat.mc 就对啦！) ：

```
1. 修改 m4 script :
[root@test root]# cd /etc/mail/
[root@test mail]# vi sendmail.mc
# 找到这几行：
dnl TRUST_AUTH_MECH(`DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl define(`confAUTH_MECHANISMS', `DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
# 将他修改成底下这样：
TRUST_AUTH_MECH(`DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
define(`confAUTH_MECHANISMS', `DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
DAEMON_OPTIONS(`Port=smtp,Addr=0.0.0.0, Name=MTA')

2. 开始制作 sendmail.cf 档案：
[root@test mail]# m4 sendmail.mc > redhat.cf
[root@test mail]# mv /etc/mail/sendmail.cf /etc/mail/sendmail.cf.bak
[root@test mail]# cp redhat.cf /etc/mail/sendmail.cf

3. 启动 sendmail 与测试 SMTP AUTH
```

```
[root@test mail]# /etc/rc.d/init.d/sendmail restart
Shutting down sendmail:                                    [  OK  ]
Starting sendmail:                                         [  OK  ]

[root@test mail]# telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 test.adsldns.org ESMTP Sendmail 8.11.6/8.11.6; Mon, 24 Feb 2003 11:51:04 +0800
ehlo localhost  <==输入本机状态的测试
250-test.adsldns.org Hello tsai.adsldns.org [127.0.0.1], pleased to meet you
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250-AUTH LOGIN PLAIN  <==出现这行就对啦!
250 HELP
quit
```

o

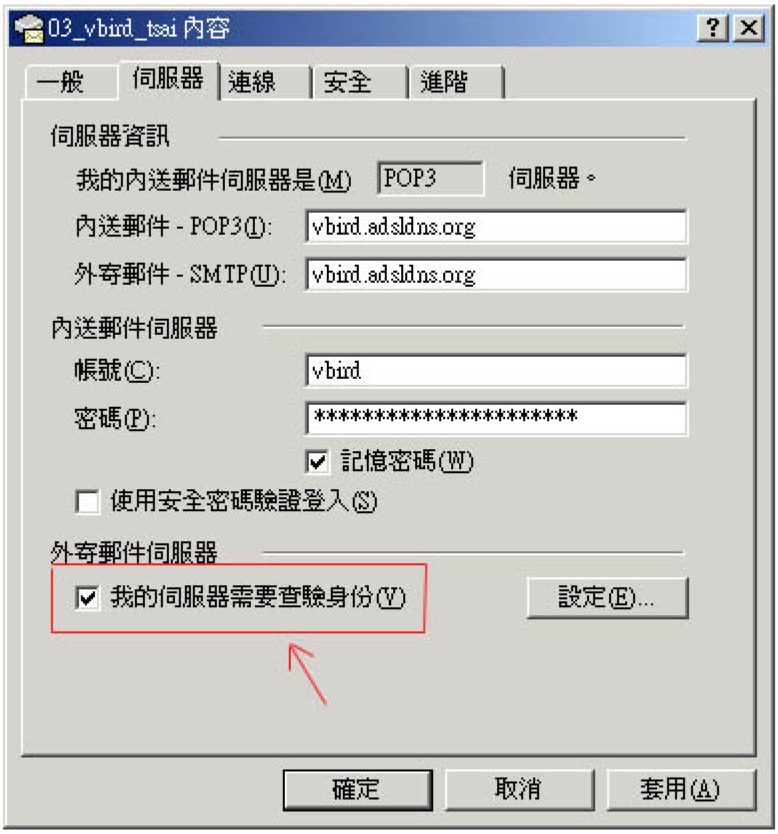
如果出现上面的 AUTH LOGIN PLAIN 那一行的话，就表示 SMTP AUTH 已经正确的启用啦！这个时候，你就可以将 /etc/mail/access 这个档案里面的相关设定给他取消掉！直接以 SMTP 来抵挡啰！不过，无论如何，如果是内部的信件，使用 /etc/mail/access 的设定还是比较简单啦！ ^_^

o

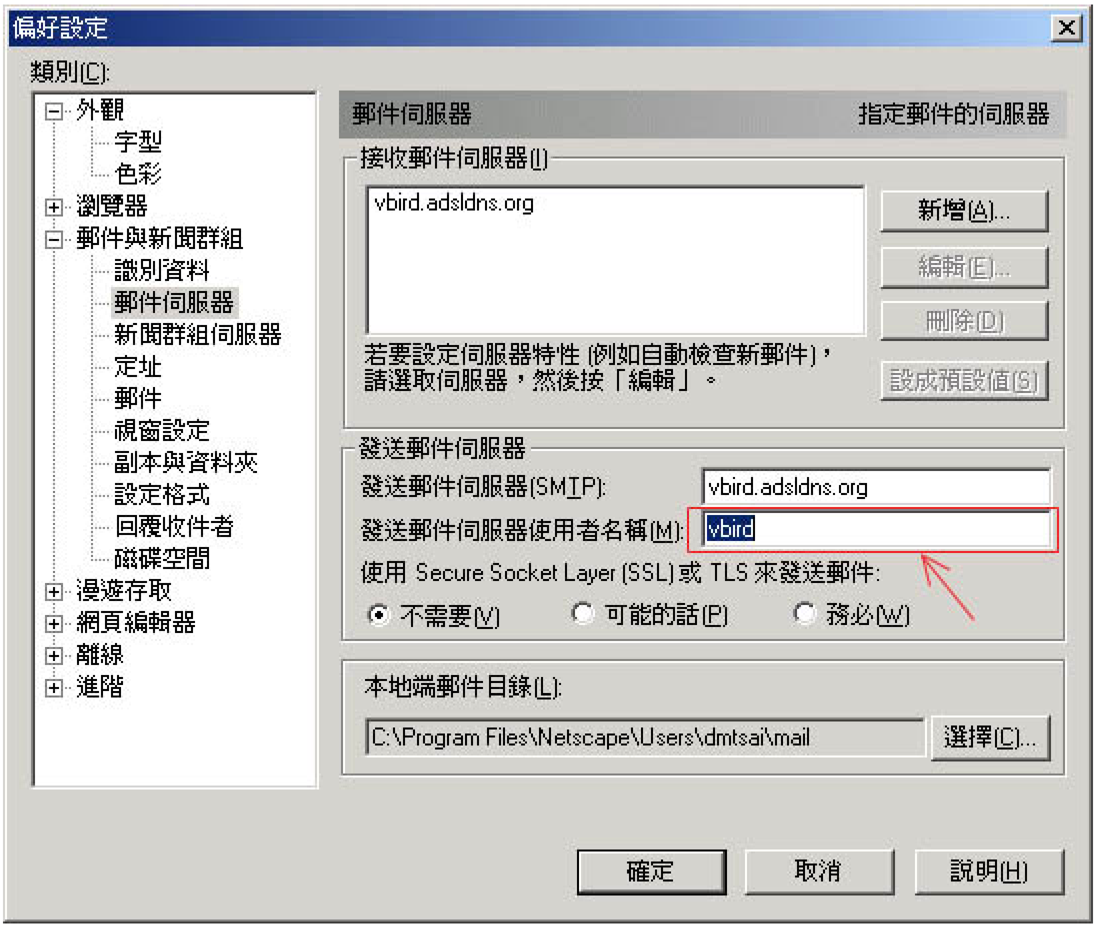
---

o 客户端 Client 的设定：

既然 SMTP AUTH 是使用使用者提供的账号密码来进行身份的确认，那么自然我们就得在 Client 的 MUA 上面进行账号密码的设定啰！如果你的系统是 Windows ，使用的 OutLook 的话，那么应该要勾选底下这个咚咚：



如果是 Netscape 的话，那就需要填写发信者姓名啰：



这样您就可以使用 SMTP AUTH 带来的方便性啰！

那么 SMTP 到底在那个步骤生效的呢？如果我们以前面提到的 一封邮件的收受流程 当中，再加入这个账号与密码的功能时，那信件会变成怎么跑呢？

12. 还是会先判断是否有收件人在本机账号中，若有则收，没有则下一步；
13. 还是会判断 /etc/mail/access.db 的动作设定，若有则作动作，否则下一步；
14. 开始启动账号密码的机制判断，若使用者提供的账号密码可以通过这个机制，则信件开始 relay ，否则下一步；
15. 将原信件退给发信者！

大致的流程就是这样啦！所以啰， /etc/mail/access 还是有存在的价值喔！尤其针对内部网域而言～

关于 ORDB 抵挡 open relay 邮件主机之机制说明与实作

什么是 Open Relay 呢？我们还记得在原理说明的地方曾经提过 Relay 这个玩意吧！目前预设的 Mail Server 都已经将 Relay 关闭了，而仅开放部分的信任网域，或者是直接经由类似 SMTP AUTH 的认证机制来进行 Relay 的功能！而如果一个 Mail Server 对于 Client 端的 Relay 需求完全的接受，那就是 Open Relay 啦！如果你的 Mail Server 被测试出有 Open Relay 的情况，那么你的邮件主机将会被当成是『黑名单』的一员，情节较轻者，可能会被停止使用 mail 的权力，情节较重者，则可能被完全停止使用网络的权力！关于台湾地区『学术网络黑名单』的 IP 数据，您可以在底下的网址查询到：

http://140.111.1.22/tanet/spam.html

我们上面介绍了很多的 Sendmail 相关的技巧与设定，可以让我们的主机不至于被当成『黑名单』的一员，不过，由于开放 Open Relay 的邮件主机对于网域内的网络具有相当大的破坏力，因为频宽会被吃光光，所以，如果我们发现有 Open Relay 的主机，其实也可以抵制我们的用户将信送到对方的主机的，以提醒我们的客户端『咦！应该要强迫对方邮件主机管理员进行维护了！』

可不可能达到这样的目的呢？可以的，就使用 Internet 上面提供的 Open Relay 名单数据库来进行『校阅』的动作啦！那么这个方法是如何工作的呢？是这样的：『当我们的 Sendmail 要传送信件到下一个 MTA 时，会先到 Open Relay 黑名单数据库查询该主机 IP 或主机名称是否在黑名单中，若不在黑名单中，则进行 Relay ，若在黑名单当中，则停止 Relay 的动作，并将信件退回，且在退回的信件上面注明此信被退回的原因，以提供使用者向网站管理员申诉的意见之用!』那么那个网络黑名单数据库在哪里呢？目前有很多黑名单数据库，鸟哥我使用的是 http://www.ordb.org 这个网站提供的数据库确认喔！

注：如果您以浏览器(Netscape 或 IE)进行 http://www.ordb.org 的浏览时，却出现一片白白的画面，这是编码的问题！请在您的浏览器上面选择编码为『繁体中文(Big5)』就可以看到该网站的内容咯！

- 如何确认一部主机是否具有 Open Relay 的问题？
  有 Open Relay 可是很危险的呢！那么如何确定我自己的主机有 Open Relay 呢？
  1. 是否已在黑名单数据库中：确认的方法很简单，直接到『http://www.ordb.org/lookup/』输入您的主机名称或者是 IP ，就可以检查是否已经在黑名单当中；
  2. 是否具有 Open Relay ：如果要测试你的主机有没有 Open Relay ，直接到『http://www.ordb.org/submit/』这个网页，同样输入主机或者 IP ，就可以将你的 IP 加入 Open Relay 的排程中，此时 ORDB 会主动的寄信给你的 Mail Server ，以确认该主机是否为 Open Relay 呢！
  3. 如何移除：如果被检查出，您的主机已经在黑名单当中，那么请立刻将 Open Relay 的功能关闭，改善你的 Mail Server 之后，再以移除的方法

『http://www.ordb.org/submit/』进行检查，如果检查出已经改善了，那就会自动将该黑名单数据库中移除！

在 Sendmail 上面启用 ORDB 网站提供的功能实在很简单，请依照底下的步骤进行即可：

```
1. 修改 m4 script :
[root@test root]# cd /etc/mail
[root@test mail]# vi sendmail.mc
# 在 MAILER 之前加入这一行，请注意，底下是『同一行』：
FEATURE(`dnsbl', `relays.ordb.org', `"Email blocked using ORDB.org - see
<http://ORDB.org/lookup/?host="$&{client_addr}">"')

2. 开始制作 sendmail.cf 档案并启动 sendmail：
[root@test mail]# m4 sendmail.mc > redhat.cf
[root@test mail]# mv /etc/mail/sendmail.cf /etc/mail/sendmail.cf.bak
[root@test mail]# cp redhat.cf /etc/sendmail.cf
[root@test mail]# /etc/rc.d/init.d/sendmail restart
Shutting down sendmail:                                [  OK  ]
Starting sendmail:                                     [  OK  ]
```

千万不要怀疑，这样即刻生效了啦！ ^_^

Procmail 相关说明

什么是 Procmail ：
什么是 procmail 呢？简单的说，Procmail 可以被视为一个 MDA 啦，也就是说，当 mail server 取得了自家的信之后，会将该封信件交给 procmail 来进行处理，而 procmail 会经由分析该邮件的内容，以及使用者设定的相对应状态，来将信件进一步过滤 (filter)！举个例子来说，你晓得有一个以『AV 情色』为邮件主旨的信件，其实他是一个广告信，那么你想让 mail server 主动的将这种信件『丢掉』的话，那么可以在 procmail 的设定档当中，加入分析此一邮件主旨，以后，若来信符合这个设定时，就会被 procmail 丢弃了！当然 procmail 的功能还不止于此，这仅是最简单的功能而已～

启用 Procmail 的支援：
基本上，如果你要启动 procmail 的话，就必需要再次的修改您的 m4 scripts 档案，也就是 redhat.mc 那个档案啦！不过， Red Hat 各个版本的 sendmail 预设都已经启动 procmail 啦！所以你可以不需要修改呢！无论如何，我们还是来看一下 与 procmail 有关的 m4 script 设定

内容吧！

```
[root@test root]# cd /etc/mail
[root@test mail]# vi sendmail.mc
# 底下仅列出与 procmail 有关的设定
define(`PROCMAIL_MAILER_PATH',`/usr/bin/procmail')dnl <==procmail 的执行档完整档名
FEATURE(local_procmail,`',`procmail -t -Y -a $h -d $u')dnl <==加入 procmail 之设定
MAILER(smtp)dnl
MAILER(procmail)dnl  <==procmail 请放置在 smtp 后面喔!
```

是的，就是这几行而已，你的 procmail 就已经被支持成为 MDA 啦！如果您是以 Tarball 安装 Procmail 的话，那么请记得将上面那几行对应到你的系统当中喔！切记切记！

开始设定 Procmail 的过滤规则：
刚刚说过， Procmail 是利用『使用者定义出的过滤规则』来进行邮件的过滤的，那么这个档案预设放在哪里呢？预设就是放在 /etc/procmailrc 这个档案啦!不过,这个档案预设是不存在的,也就是说，虽然你启动的 procmail 的 MDA 功能，但是你并没有启用任何的规则，所以当然进入 mail server 的邮件，就不需要被『过滤』啦！而如果你定义出规则在 /etc/procmailrc 之后，以后信件就会拿该规则来进行过滤啰！在此需要先介绍一下，一封邮件的内容主要分为两部分：

17. 表头 (Header) 一些基本信息如寄件者、标题等等;
18. 主要内文 (Body) 部分

至于 Procmail 过滤规则的设定方法是这样的：

```
一组规则设定至少共含有三行：
:0 [flags] [ : [locallockfile] ]  <==设定要被过滤邮件的检查地方
<zero or more conditions (one per line)>  <==被检查的规则内容
<exactly one action line>  <==若符合所定义的规则内容，则邮件要进行的动作！

参数说明：
a. 关于检查邮件的部分(第一行)，flags 包含有：
    H : Header 的检查
    B : Body 的检查
    h : 提供 Header 的数据，进入 pipe 、 file 及 mail 等的检查
    b : 提供 Body 的数据进入 pipe, file 及 mail 等的检查！
b. 关于检查内容的部分：
    ! : 这是反向选择的意思
    < : 检查 mail 的总长度是否小于设定值 ( bytes )
    > : 与 < 相反的选择啰！
c. 关于动作的部分
    | : 开始启用后续的 shell 程序工作！
d. 其它的环境变量部分：
    PATH        搜寻执行文件的路径
```

```
    SENDMAIL     那个 /usr/sbin/sendmail 啰!
    LOGFILE     登录档! 通常设定在 /var/log/procmail.log
e. 与 Regular express 相关的部分:
    ^ : 开始(同一行最左边)字符
    $ : 本行的结束字符(最右边)的比对
    . : 除了新增一行之外的任何字符!
    \ : 跳脱字符
更详细的内容请务必参考:
man procmailrc
man procmailex
```

如果有兴趣的话，可以参考我设定的 /etc/procmailrc 的档案内容喔! 你也可以下载 (http://linux.vbird.org/download/index.php#sendmail_sendmail)喔!

```
# History of procmail RC file,
# Written by VBird
############ History ############################################
# Day  Who What
# ---------- ------ ---------------------------------------
# 2002/05/04 VBird First build, all is the same with
#    卧龙小三 ftp://ftp.tnc.edu.tw/pub/Sysop/MAIL/procmailrc
#    林克敏   http://freebsd.lab.mlc.edu.tw/procmail.htm
# 2002/05/04 VBird 新增广告信件移除两个!
# 2002/05/06    VBird   新增 Klez.W32 病毒的针测
# 2003/09/05 VBird 新增 Sobig 与 MiMail 病毒抵挡!
#################################################################
############ Globle settings ####################################
LOGFILE=/var/log/procmail.log
UMASK=000

################ Virus scanner ##################################
############# 1. KLEZ.G Virus ###################################
:0b
* ^Subject:.*(Let's be friends)
/dev/null

:0b
* ^Subject:.*A funny game
/dev/null

:0b
* ^Subject:.*Hello\,.*\,how are you.*
/dev/null

```

```
:0 B
* ^Content-Type:.*audio/x-wav.*
* ^.*name=.*\.(scr|SCR)
/dev/null

:0 B
* ^Content-Type:.*audio/x-midi.*
* ^.*name=.*\.(scr|SCR)
/dev/null

:0 B
* ^Content-Type:.*application/octet-stream.*
* ^.*name=.*\.(scr|SCR)
/dev/null

:0 Bb
* ^This game is my first work.*
* ^You\'re the first player.*
* I.*you would .* it.*
/dev/null

:0 Bb
* Hello,This is a.*
* I.*you would.*it.*
/dev/null

:0 Bb
* ^This is a.*
* ^I.*you would.*it.*
/dev/null

:0 Bb
* .*This is a.*patch.*
* ^I .* you would.*it.*
/dev/null

:0 B
* ^Content-Type:.*multipart/mixed.*
* name="ANTI_CIH.EXE"
/dev/null

:0b
* ^Subject:.*W32.*removal tools$
/dev/null
```

```
:0b
* ^Subject:.*Worm Klez.*immunity.*
/dev/null

############# 2. Nimda Virus #####################################
:0 Bh
* ^Content-Type:.*audio/x-wav.*
* name="readme.exe"
/dev/null

:0 Bh
* ^Content-Type:.*audio/x-wav.*
* name="sample.exe"
/dev/null

:0 B
* ^Content-Type:.*multipart/mixed.*
* name="readme.exe"
/dev/null

:0 B
* ^Content-Type:.*multipart/mixed.*
* name="sample.exe"
/dev/null

:0 B
* ^Content-Type:.*application.*
* name="readme.exe"
/dev/null

:0 B
* ^Content-Type:.*application.*
* name="sample.exe"
/dev/null

:0 Bh
* charset="iso-8859-1"
* name=.*bat
/dev/null

:0 Bh
* charset="iso-8859-1"
* name=1.*zip
/dev/null
```

```
############# 3. SirCam Virus ###################################
:0 Bh
* I send you this file in order to have your advice
/dev/null

############# 4. MMail  Virus ###################################
:0 Bh
* name=.*message.zip
/dev/null

############# 5. Sobig  Virus ###################################
:0 Bh
* name=.*pif
/dev/null


#################################################################
############# 广告信  ###########################################

# 来自奇摩与 PCHOME 的都挡掉!
:0 Bh
* href\=.*http:\/\/home\.kimo\.com\.tw
/dev/null

:0 Bh
* href\=.*http:\/\/home\.pchome\.com\.tw
/dev/null

:0 Bh
* Subject:.*AV66 情色派
/dev/null

:0 Bh
* From:.*创意营销
/dev/null

:0 Bh
* http:\/\/.*etl\.com\.tw
/dev/null

:0 BhH
* http:\/\/f2m\.aac\.com\.tw
/dev/null

:0 BhH
```

```
* http:\/\/www\.9895\.com
/dev/null

:0 BhH
* http:\/\/88\.to
/dev/null

:0 BhH
* http:\/\/fly\.to
/dev/null

:0 BhH
* http:\/\/china\.kyodo\.co\.jp
/dev/null

:0 BhH
* http:\/\/.*\.openmybiz\.com
/dev/null

:0 BhH
* http:\/\/.*\.pufan\.com
/dev/null

:0 BhH
* http:\/\/.*\.buyy\.biz
/dev/null

:0 BhH
* http:\/\/.*\.tw886\.to
/dev/null

:0 BhH
* http:\/\/.*\.hala\.idv\.tw
/dev/null

# 底下是最近的一些发信软件所发送的几个 Keyword. 2003/09/05
:0 Bh
* http:\/\/.*\.wantclick\.com
/dev/null

:0 Bh
* http:\/\/fastcounter\.bcentral\.com
/dev/null
```

此外，您还要新增一个档案在 /etc/logrotate.d 里面才行！因为你会多一个

/var/log/procmail.log 的档案啊！

```
[root@test root]# vi /etc/logrotate.d/procmail
/var/log/procmail.log {
        monthly
        rotate 5
        nocompress
}
```

其它 Procmail 相关的信息可以查询底下的网站喔！
http://www.procmail.org
http://www.sektorn.mooo.com/era/procmail/mini-faq.html

Tarball 的安装方式（ 适用于原本 Linux 没有 sendmail 或者是认证机制的！ ）
Sendmail 的安装方面真的是相当的『雪特！』比那个 LAMP 还要麻烦的多～所以，如果你的系统当中主要是以 sendmail 做为你的邮件服务器软件，例如 Red Hat 7.2, Red Hat 7.3 等等的，那么就以你的 distribution 提供的软件来安装！不要直接使用 Tarball 的方式安装，因为要配合的套件实在太多了，鸟哥也说不准是否有什么咚咚忘记给他安装上来ㄋㄟ！所以，除非你是最近释出的 distribution 例如 Mandrake 9.0 ，因为他使用的并非是 sendmail ，或者是 OpenLinux Server 3.1.1 ，因为他的 sendmail 预设没有 认证机制 存在，否则，尽量以你的 distribution 提供的 sendmail（ RPM 版本 ）来安装你的邮件服务器吧！例如 Red Hat 7.x 版的就直接由 Red Hat 提供的 sendmail 为准吧！

如果你是使用 OpenLinux Server 3.1.1 以及 Mandrake 或者是其它并非使用 sendmail 的 Linux distribution 时，由于这些 distribution 不是提供其它的 mail server package 而没有 sendmail ，就是在邮件服务器套件上面缺东缺西的，例如缺乏 procmail, cyrus-sasl....等等的套件，所以，看来您也只好摸摸鼻子，好好的自行加油努力以 tarball 的方式来安装啰！那么与 RPM 版本类似的，你需要的套件有哪些呢？至少需要有底下这几个，邮件服务器的功能才够完整喔：

- sendmail：http://www.sendmail.org/
- cyrus-sasl：http://asg.web.cmu.edu/cyrus/download/
- procmail：http://www.procmail.org

基本上，我们目前仅安装 sendmail 及 cyrus-sasl 就好了！不过，考虑有些 Linux Distribution 并没有提供相关的功能，所以我们还是安装一下 procmail 好了！因为那个 Cyrus SASL 就是要达成 SMTP 认证的重要工具，而我们的 senmdail 是用他来做认证的，所以两者要一起编译与安装才行呐！！直到目前为止（ 2003/02/20 ）我们使用最新的 Cyrus-SASL 2.1.12 版以及 procmail 3.22，至于 sendmail 则使用 8.12.7 版！首先，我们必需先建立密码数据的函式库，亦即 Cyrus SASL 这个套件与 Procmail 喔：（ 注：目前 sendmail 或其它的，例如 postfix 等邮件服务器软件，主要仍以 Cyrus-sasl 1.5.28 这个版本来安装的，包括 sendmail 官方网站的介绍「http://www.sendmail.org/~ca/email/auth.html」亦是使用 1.5.28 来做介绍！不过， Cyrus SASL 的官方网站说， 2.xx 版本的 SASL 比较优良，所以，这里我们参考了 sendmail 官方网站，以及 Cyrus SASL 的官方网站的文章，整理出底下的安装步骤啰！ ）

安装 Cyrus SASL 2.xx 版本！

```
1. 首先将数据解压缩(假设您将我们网站的档案捉到 /root 底下了！)
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/cyrus-sasl-2.1.12.tar.gz
.....(略).....最后建立一个 cyrus-sasl-2.1.12 的目录

2. 再来就是设定你的 cyrus SASL 的参数以及编译啰!
[root@test src]# cd cyrus-sasl-2.1.12
[root@test cyrus-sasl-2.1.12]# ./configure   \
> --prefix=/usr/local/cyrus-sasl2 --enable-login --enable-plain \
> --enable-pwcheck --with-saslauthd=/var/run
# 注意上面的语法！！那个 \ 是跳脱字符喔！后面直接 Enter ！
# 不要接任何空白或者是 tab 按键！若有问题，请到 Shell 看看！
# 至于那个 pwcheck 的项目，就是要用来做为密码确认的一个咚咚啰！

[root@test cyrus-sasl-2.1.12]# make clean && make
[root@test cyrus-sasl-2.1.12]# make install
# 上面三个步骤会花去一些时间，请耐心等待～
# 而由于我们刚刚设定了 --prefix=/usr/local/cyrus-sasl2 这个参数，
# 所以， make install 之后，我们有用的函式库会在
# /usr/local/cyrus-sasl2/lib/sasl2 这个路径当中，但是， cyrus 程序
# 会主动去找 /usr/lib/sasl2 这个目录！所以，我们必需要做连结档！

[root@test cyrus-sasl-2.1.12]# cd /usr/lib
[root@test lib]# ln -s /usr/local/cyrus-sasl2/lib/* .
# 这样就建立起连结档啰！很厉害喔！

3. 准备建立 sendmail 与 cyrus SASL 使用的简易设定档：
[root@test lib]# cd /usr/lib/sasl2
[root@test sasl2]# echo 'pwcheck_method: saslauthd' > Sendmail.conf
# 注意了！一般来说， sendmail 会使用 SASL 这个函式库里面，
# 在 /usr/lib/sasl2/Sendmail.conf 这个档案的设定做为他的预设使用 SASL 的
# 参数档案，那个 Sendmail 的 S 是大写，请不要写错了！我们使用 SASL 预设的
# saslauthd 这支程序做为密码认证的 daemon 啰!

4. 建立一些需要的参数：
[root@test sasl2]# vi /etc/man.config
# 新增一行：
MANPATH /usr/local/cyrus-sasl2/man

5. 检验 saslauthd 这支程序是否可行!
[root@test sasl2]# /usr/local/cyrus-sasl2/sbin/saslauthd -a shadow
[root@test sasl2]# cd /usr/local/src/cyrus-sasl-2.1.12/saslauthd/
[root@test saslauthd]# make testsaslauthd
[root@test saslauthd]# ./testsaslauthd -u userID -p 'yours.passwd'
```

```
0: OK "Success."
# 若显示 OK 的话！那么就是成功啦！很好！我喜欢～

6. 设定开机时启动
[root@test saslauthd]# vi /etc/rc.d/rc.local
# 加入这一行：
/usr/local/cyrus-sasl2/sbin/saslauthd -a shadow
```

```
                              安装 Procmail
1. 首先将数据解压缩(假设您将我们网站的档案捉到 /root 底下了！)
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/procmail-3.22.tar.gz
.....(略).....最后建立一个 procmail-3.22 的目录

2. 直接给他安装！
[root@test src]# cd procmail-3.22
[root@test procmail-3.22]#  make install
[root@test procmail-3.22]#  which procmail
/usr/bin/procmail
```

这样就大功告成了！接下来准备安装 Sendmail 啰！

```
0. 首先将原有的 sendmail 数据备份例如 /etc/mail 目录底下的档案！
   并且，必需要确定有安装 Berkeley DB 等相关的函式库！这是 makemap 要用的喔！
[root@test root]# mv /etc/mail /etc/mail.old
# 请注意！如果您使用的 Linux Distribution 并非为 sendmail ，那么上面的
# 目录可能不会存在！请详细的参考您的主机上面的说明喔！

[root@test root]# locate libdb.so
/lib/libdb.so
# 这个档案请『务必存在！』可能是 libdb.so 或 libdb.a 的形式！
# 这个档案是 Berkeley 数据库的主要函式库，这是用来做为 makemap
# 的一些数据库格式所必需要的函式库，如果你的系统不存在，那么请
# 拿出你的原版光盘片来安装！通常这个套件的档名应该是 db3-devel.....
# 或者是 libdb3..... 的文件名称！请以 RPM 来安装吧！
# 基本上，反正你的光盘片上(不管几片，全部拿出来喔！)所有
# 档名为 db#### 及 libdb### 的档案都装上去就对了！
# 至于相关的属性相依问题，请参考 RPM 与 Tarball 的安装一文。

[root@test root]# locate libwrap
/usr/lib/libwrap.a
# 这个档案与等一下我们要建立的 TCP_Wrappers 的支持有关，
# 请确定他的存在喔！如果不存在的话，而你的系统又是 MDK 9.0 时，
```

```
# 可以拿出第三片来安装 tcp_wrappers-devel..... 那个 RPM 档案，
# 如果是其它的 distribution 同时又找不到 libwrap.a 时，
# 那底下 site.config.m4 里面的 -DTCPWRAPPERS 及 -lwrap 都拿掉！
# 不要支持也没有关系！

1. 将 sendmail 解压缩，假设您下载的数据在 /root 底下
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/sendmail.8.12.7.tar.gz
.....(略).....会建立一个名为 sendmail-8.12.7 的目录

2. 建立 sendmail 所需要的 Cyrus SASL 支持之设定项目：
[root@test src]# cd /usr/local/src/sendmail-8.12.7/devtools/Site
[root@test Site]# vi site.config.m4
PREPENDDEF(`confMAPDEF', `-DMAP_REGEX')
APPENDDEF(`confENVDEF', `-DTCPWRAPPERS -DSASL=2')
APPENDDEF(`conf_sendmail_LIBS', `-lwrap -lsasl2')
APPENDDEF(`confLIBDIRS', `-L/usr/local/cyrus-sasl2/lib')
APPENDDEF(`confINCDIRS', `-I/usr/local/cyrus-sasl2/include')
define(`confMANROOT', `/usr/share/man/man')
# 这个 site.config.m4 是 sendmail 在编译的时候会主动去读取的主机自行设定文件。
# 上面这六行是需要自行建立的！请特别注意， cyrus sasl 的 2.x 与 1.5 版
# 在安装与设定上是不一样的！所以这里请特别留意喔！不要搞错了！
# 第一行：上面第一行与一些正规表示法有点关系，可加可不加！
# 第二行：第二行在设定支持的模式，我们支持了 TCP_Wrappers 及 SASL 身份认证！
#         关于 TCP_Wrappers 请参考前面『简易防火墙』的说明吧！
#         至于 -DSASL=2 则是 SASL 第二版的格式！
# 第三行：第三行与第二行有点关系，使用 TCP 时需要用到 libwrap.a 这个档案，
#         那就是 lwrap 这个标志！至于 -lsasl2 就是 libsasl2.so 那个档案啦！
# 第四行与第五行：这两行 /usr/local/cyrus-sasl2 指的是我的 SASL 的数据库所在目录
# 如果你不是安装在这个目录的话，请依照您刚刚在建立 cyrus sasl 下达的
# --prefix=/你的/目录来填写喔！还有其它相关的说明，请参考您主机内的
# /usr/local/src/sendmail-8.12.7/sendmail/README
# 请注意，如果您确定可以支持 TCP_Wrappers 之后，那么你就可以在
# /etc/hosts.deny, /etc/hosts.allow 以底下的样式来抵挡 IP 或主机名称
sendmail: 192.168.0.0/255.255.255.0 :Allow
sendmail: 192.168.0.100: deny
# 更多的 TCP_Wrappers 信息请参考『认识网络安全』

3. 开始编译 sendmail 啰与新增 sendmail 管理员 smmsp
[root@test Site]# cd /usr/local/src/sendmail-8.12.7/sendmail
[root@test sendmail]# sh Build -c
.....(略).....会花很多时间喔！
[root@test sendmail]# groupadd -g 40 smmsp
[root@test sendmail]# useradd -M -g smmsp -u 40 -d /var/spool/clientmqueue  \
```

```
>  -s /dev/null smmsp
[root@test sendmail]# mkdir -p /var/spool/clientmqueue
[root@test sendmail]# chown -R smmsp:smmsp /var/spool/clientmqueue
[root@test sendmail]# chmod -R 770 /var/spool/clientmqueue
# 新增一个使用者，他无法登入，ID 是 40 号！ 这个使用者的家目录是 /var/spool/clientmqueue
# 主要仅用于邮件的收受与传递！增加这个使用者是 sendmail 8.12 板后新增的功能！
# 主要的目的在于提供更安全的 sendmail 使用环境！

4. 设定 macro 档案
[root @test sendmail]# cd /usr/local/src/sendmail-8.12.7/cf/cf
[root @test cf]# vi sendmail.mc
# 这个档案的内容是你必需要建立的！如果你要跟我一样的话，就用底下的设定吧！
divert(-1)
dnl ===========================================================================
dnl     This file is modified from Red Hat 7.2's redhat.mc file.  VBird 2003/02/20
dnl     The functions of sendmail are as following
dnl            dnl                    ==> Just mark, such as # in shell scripts
dnl             VERSIONID             ==> The version of sendmail and vender
dnl             OSTYPE                ==> The Operation System type
dnl             define                ==> difine some usefull functions
dnl             FEATURE               ==> some functions and files' location !
dnl             other settings        ==> Other settings.
dnl             MAILER                ==> mail protocol and featurs.
dnl
dnl     The following is the command to macro the *.mc to *.cf !
dnl
dnl             sh Build sendmail.cf
dnl             cp sendmailcf /etc/mail
dnl
dnl ======1. Some informations ================================================
include(`../m4/cf.m4')
VERSIONID(`Sendmail for Linux using Mandrake 9.0')
OSTYPE(`linux')
define(`confDEF_USER_ID',``8:12'')

dnl ======2. Some settings ====================================================
define(`confTO_CONNECT', `1m')dnl The timeout waiting for an initial connect (1 minute here)
define(`confTRY_NULL_MX_LIST',true)dnl If this host is the best MX for a host and other
arrangements haven't been made, try connecting to the host directly; normally this would be
a config error.
define(`confDONT_PROBE_INTERFACES',true)dnl About /etc/mail/mailertables !
define(`ALIAS_FILE', `/etc/mail/aliases')dnl About username aliases
define(`STATUS_FILE', `/etc/mail/statistics')dnl
define(`UUCP_MAILER_MAX', `2000000')dnl
```

```
define(`confUSERDB_SPEC', `/etc/mail/userdb.db')dnl
define(`confPRIVACY_FLAGS', `authwarnings,novrfy,noexpn,restrictqrun')dnl

dnl ======3. About some other important settings ==================================
define(`PROCMAIL_MAILER_PATH',`/usr/bin/procmail')dnl About procmail settings
define(`confAUTH_OPTIONS', `A')dnl The following three lines are about SASL settings
TRUST_AUTH_MECH(`LOGIN PLAIN')
define(`confAUTH_MECHANISMS', `LOGIN PLAIN')

dnl ======4. important settings here! =============================================
FEATURE(`no_default_msa',`dnl')dnl Don't generate the default MSA daemon,
DAEMON_OPTIONS(`Port=smtp,Addr=0.0.0.0, Name=MTA')dnl This must be set because the up line
FEATURE(`mailertable',`hash -o /etc/mail/mailertable.db')dnl
FEATURE(`virtusertable',`hash -o /etc/mail/virtusertable.db')dnl
FEATURE(redirect)dnl About Redirect the address to another one mail server !
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl useing /etc/mail/local-host-names
FEATURE(use_ct_file)dnl useing /etc/mail/trusted-users
FEATURE(local_procmail,`',`procmail -t -Y -a $h -d $u')dnl about procmail
FEATURE(`access_db',`hash -T<TMPF> -o /etc/mail/access.db')dnl
FEATURE(`blacklist_recipients')dnl
EXPOSED_USER(`root')dnl
FEATURE(`accept_unresolvable_domains')dnl

dnl ======5. About ORDB deny the open relay mail servers ==========================
FEATURE(`dnsbl', `relays.ordb.org', `"Email blocked using ORDB.org - see
<http://ORDB.org/lookup/?host="$&{client_addr}">"')

dnl ======6. MAILER settings here ================================================
MAILER(smtp)dnl
MAILER(procmail)dnl
Cwlocalhost.localdomain
# 重点是上面四行粗体字喔！特别留意了！

5. 开始给他制作 sendmail.cf 这个档案啰！
[root@test cf]# sh Build sendmail.cf <==这个在制作 sendmail.cf 这个档案
[root@test cf]# mkdir -p /etc/mail
[root@test cf]# sh Build install-cf
# 这个在安装 sendmail.cf 到 /etc/mail 底下去！

6. 开始安装 sendmail 主程序，以及其它相关的程序，例如 makemap 等等的！
[root@test cf]# cd /usr/local/src/sendmail-8.12.7/sendmail/
[root@test sendmail]# sh Build install
[root@test sendmail]# cd ../makemap
```

```
[root@test makemap]# sh Build install
[root@test makemap]# cd ../mailstats
[root@test mailstats]# sh Build install

7. 其它档案的建立与修订!
[root@test sendmail]# cd /etc/mail
[root@test mail]# echo 'test.adsldns.org' >> local-host-names
# 这里请输入你的主机名称
[root@test mail]# echo 'localhost    RELAY'  >> access
[root@test mail]# makemap hash access < access
[root@test mail]# touch domaintable
[root@test mail]# makemap hash domaintable < domaintable
[root@test mail]# touch mailertable
[root@test mail]# makemap hash mailertable < mailertable
[root@test mail]# touch trusted-users
[root@test mail]# touch virtusertable
[root@test mail]# makemap hash virtusertable < virtusertable
[root@test mail]# mkdir -p /var/spool/mqueue
[root@test mail]# chown root:wheel /var/spool/mqueue/
[root@test mail]# chmod 700 /var/spool/mqueue
[root@test mail]# touch aliases
[root@test mail]# sendmail -v -bi
/etc/mail/aliases: 0 aliases, longest 0 bytes, 0 bytes total
# 若是出现上面的字样的话(不一定是这样的!但反正就是不会显示错误讯息就是了!)
# 就表示您的 sendmail 应该已经『没问题啦!』
[root@test mail]# sendmail -bd -q30m   <==启动 sendmail 看看吧!
[root@test mail]# telnet localhost 25 <==试看看连的上吗?!
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
220 test.adsldns.org ESMTP Sendmail 8.12.7/8.12.7; Tue, 18 Feb 2003 21:56:00 +0800
ehlo localhost   <==这里输入测试列!
250-test.adsldns.org Hello localhost.localdomain [127.0.0.1], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-DELIVERBY
250 HELP
quit  <==这里输入离开字样!
221 2.0.0 test.adsldns.org closing connection
```

```
Connection closed by foreign host.
# 如果看到出现上面的那行黄色加粗字体，呵呵！那就是 OK 啦！
```

上面的 site.config.m4 及 sendmail.mc 可以在鸟哥的网站上面下载喔 (http://linux.vbird.org/download/index.php#sendmail_sendmail)！免得打错字.....基本上，Tarball 的安装方式有点麻烦，此外，又容易发生问题，所以，除非必要，否则建议还是使用原本 Linux distribution 所提供的 RPM 或者其它的套件管理员的方式来安装吧！此外，关于 aliases 与启动 sendmail 的 script 也可以在上面的相关连结下载喔！安装方式中，需要注意的是在您下载完毕之后，请：

1. Procmail 如果本机上面已经安装了，不需要重新移除后再安装！直接使用本机原有的即可！
2. 依照上面的方法安装所有的套件；
3. 将 aliases 放置在 /etc/mail 底下；
4. 将 procmailrc 放置在 /etc/ 底下；
5. 在 /etc/logrotate.d 增加 procmail 的 logrotate 数据；
6. 将 sendmail 放置在 /etc/rc.d/init.d/ 底下
7. 在 /etc/rc.d/rc.local/ 最底下新增一行即可开机时启动：
   /etc/rc.d/init.d/sendmail start

这样就已经安装好整个 Sendmail 啰！设定上是很容易啦！不过，维护上面可就要多费心啰！

其它应用说明：
上面的 Sendmail 说明，如果您都已经详细的参详之后，应该对于你的 Mail Server 管理的能力具有蛮强的认知了！不过，系统管理员无时无刻总是会遇到突发状况的！所以，这里我们来聊一聊一些可能会发生在你我身上的 Mail Server 可能会遇到的问题吧！

无法寄信时的可能问题说明与解决之道

虽然 Mail 很方便，但是仍然会有无法将信件寄出的时候！如果您已经设定好 Sendmail 了，但是总是无法将邮件寄出去，那可能是什么问题呢？

1. 关于硬件配备：无论任何情况之下，如果硬件出问题，那么所有的服务都将不正常啦！所以，请先检查您的硬件是否『怪怪的！』这个先确认 OK 吧！
2. 关于网络参数的问题：如果连不上 Internet ，那么哪里来的 Mail Server 呢？所以请先确认你的网络已经正常的启用了！关于网络的确认问题，请查阅前几篇『Linux 网络侦错』的内容介绍；
3. 关于服务的问题：请务必确认 port 25 与 port 110 已经正确的启动了！使用 netstat 指令即可了解是否已经启动该服务！
4. 关于防火墙的问题：很多时候，很多朋友使用 Red Hat 或者其它 Linux distribution 提供的防火墙设定软件，结果忘了启动 port 25 与 port 110 的设定，导致无法收发信件！请特别留意这个问题喔！可以使用 iptables (核心为 2.4.xx 版本) 或者是 ipchains (核心为 2.2.xx 版本) 来检查是否已经启用该 port 呢！其余请参考简易防火墙设定那一章喔！
5. 关于 TCP_Wrappers 的问题：如果你的 sendmail 还有支持 tcpd 这个程序 (或者说是 libwrap 这个函式库) 的话，那么在 /etc/hosts.allow 与 /etc/hosts.deny 的设定

也会影响到收发信件的正常与否！如果 /etc/hosts.deny 有设定『ALL: ALL』的话，那么请务必在 /etc/hosts.allow 里面加设『 sendmail: ALL 』喔！

6. 关于设定档的问题：在 sendmail 8.11 版本中，仅有 sendmail.cf 这个设定档，但是在 8.12 当中多了个 submit.cf 这个寄信功能档案，请务必确定您的 *.mc 设定是正确的！说说鸟哥的经验，有一次在测试的时候，一直发信无法将信件寄出去！明明 sendmail.mc 都没有什么问题，就是 local 无法寄信！后来才发现，因为我更动过 submit.mc 这个档案，重新以 m4 跑过之后，忘记将他给改回来了！结果花了我两天的时间在搞 sendmail.mc ....所以，在作任何一步动作的时候，请千万记住『作笔记』或者『将动作记下来！』
7. 关于档案权限的问题：一般来说，如果以 RPM 安装 sendmail 会比较没有问题，而如果以 Sendmail 8.12 以后版本手动安装的话，那由于 sendmail 对于安全的要求越来越严格，所以你必须针对每个目录或档案进行检查才行！通常检查的目录为：
    - /etc/mail ：里面的档案至少都为 644 或 640 的权限！
    - /var/spool/mqueue ：务必为 700 的权限
    - /var/spool/clientmqueue：这在 8.12 才有，所有人与群组务必为 smmsp ，而权限务必为 770 ；
    - 每个 ./forward 的档案需要控制其权限喔！在 8.12 版本中，.forward 的『拥有群组』必须要为 smmsp ，并且其权限必须要为 640 才行喔！
8. 关于使用者的设定问题：一般而言，如果使用者不登入 sendmail 主机进行寄信的动作(local mailer)，那么 /etc/passwd 里面的设定就无关紧要了！不过，如果该使用者想要在 sendmail 本机上面使用 mail 的功能，那么在 8.12 版本当中，您就必须要：
    - 使用者的 primary 群组必须要为 smmsp ！
    - 使用者的 shell 必须要可以登入才行！
    - 其它使用者的相关档案当中，最明显的 .forward 权限必须设定正确！
9. 其它档案的设定问题：
    - 如果发现只有某个 domain 可以寄信，其它的同一主机的 domain 无法寄信，需要检查 local-host-names 这个档案的设定；
    - 如果发现邮件被挡下来了！而且老是显示 reject 的字样，那么可能被 /etc/mail/access 挡住了；
    - 如果发现邮件队列 (mailq) 存在很多的邮件，可能是 DNS 死掉了，请检查 /etc/resolv.conf 的设定是否正确！
10. 其它可能的问题：最常发生的就是认证的问题了！这是由于使用者没有在 MUA 上面设定『我的邮件需要认证』的选项啦！请叫你的 client 端用户赶紧勾选吧！
11. 还是不知道问题的解决方案：一般而言，上面的几个讯息应该可以提供您校正 sendmail 的问题了，不过，如果还是查不出问题的话，那么请务必检查您的 /var/log/maillog (有的时候是 /var/log/mail ，这个要看 /etc/syslog.conf 的设定)，当你寄出一封信的时候，例如 vbird 寄给 bird2@tsai.adsldns.org 时，那么 maillog 档案里面会显示出两行，一行为 from vbird一行为 to bird2@tsai.adsldns.org，也就是『我由哪里收到信，而这封信会寄到哪里去！』的意思，由这两行就可以了解问题了！尤其是 to 的那一行，里面包含了相当多的有用信息，包括邮件无法传送的错误原因的纪录！如果您对于登录档不熟，请拿出『鸟哥的 Linux 私房菜 -- 基础学习篇』里面的『认识登录档』一文吧！(注：这就是鸟哥为什么老是希望大家能够先看完基础篇的原因，太重要了！)

关于备份

不管什么时候，备份总是重要的！那么如果我是单纯的 Mail Server 而已，我需要的备份数据有哪些呢？

1. /etc/procmailrc 这个档案；
2. /etc/passwd, /etc/shadow, /etc/group 等与账号有关的资料；
3. /etc/mail 底下的所有档案数据；
4. /etc/sendmailcf 或者 /etc/aliases 等等 sendmail 相关档案(因为可能不放在 /etc/mail 当中！)
5. /home 底下的所有使用者数据；
6. /var/spool/mail 底下的档案与 /var/spool/mqueue 邮件队列档案；
7. 如果是 Sendmail 8.12 则可以考虑储存 /var/spool/clientmqueue。

如果真的仅要备份这些资料的话，我可以写一支程序让他每周备份一次喔！假设该程序可以放置在 /usr/local/backup/backup.sh 这里，并且备份的数据也都放置在此！当然，可以的话，应该是要放置在另一颗硬盘，甚至是另一个储存装置，例如 tape 等等比较好的啦！底下提供一个程序范例啰：

```
#!/bin/bash
# 这支程序可以用来备份 mail server 的账号资料喔!
# 撰写者 VBird 2003/02/24

# 0. 设定目录与相关的变量:
dir=/usr/local/backup
[ -z "$dir" ] || mkdir -p "$dir"
[ -f "$dir" ] && echo "$dir exist, but is not a directory, stop here" && exit

# 开始备份一些档案:
cp -a /etc/passwd   $dir
cp -a /etc/shadow   $dir
cp -a /etc/group    $dir
[ -z /etc/procmailrc ]  && cp -a /etc/procmailrc $dir
[ -z /etc/sendmail.cf ] && cp -a /etc/sendmail.cf $dir
[ -z /etc/aliases ]     && cp -a /etc/aliases

# 开始备份目录:
tar -zcvf $dir/home.tar.gz     /home
tar -zcvf $dir/etcmail.tar.gz  /etc/mail
tar -zcvf $dir/varmail.tar.gz  /var/spool/mail
tar -zcvf $dir/mqueue.tar.gz   /var/spool/mqueue
[-z /var/spool/clientmqueue ] &&  \
tar -zcvf $dir/clientmqueue.tar.gz /var/spool/clientmqueue
```

你可以在网站下载(http://linux.vbird.org/download)然后将这支程序改变一下属性『chmod 755

backup.sh』之后，放到 crontab 里面去执行就可以啦！

关于 quota 的设定与 /var/spool/mail 目录的转移

网络上有很多『免费的电子邮件信箱』空间，一般而言，使用的就是 quota 这个磁盘配额工具！因为我们的 Linux 主机硬盘空间就是这么多！当然啰，使用磁盘配额 (quota) 会是一个对大家比较公平的方法！使用 quota 的技巧已经在『鸟哥的 Linux 私房菜 -- 基础学习篇』里面介绍过了，这里不再重复介绍，要介绍的是几个可能会发生在实际的案例中的一些小技巧：

- 邮件信箱所在的磁盘空间不足了：
  这是很可能会发生的问题啊！尤其是在用量很大的网站上面！这个时候你的解决方法主要有：
  1. 新增加一颗硬盘，格式化好之后将他 mount 到 /var/spool/mail 这个目录下；
  2. 如果主机里面还有其它目录具有很大的空间，例如 /home 这个地方，那么就可以：
     cd /var/spool
     mv mail /home
     ln -s /home/mail mail

- 使用 quota 设定：
  一般而言，我们通常会将 /home 做为 quota 的 partition ，那么 /var/spool/mail 其实也可以依附在 /home 这个 partition 之下，来达到 quota 对于使用者的规范喔！达成的方法很简单啦：
  1. 先在主机规划与安装的时候，让 /home 独立于一个 partition 当中；
  2. 以『鸟哥的 Linux 私房菜 -- 基础学习篇』的 quota 内容为范例，建立好 /home 的 quota 限额；
  3. 将 /var/spool/mail 整个搬到 /home 底下，并做好连结的动作就可以立即生效啦：
     cd /var/spool
     mv mail /home
     ln -s /home/mail mail

- 关于使用者邮件的放置地点：
  很多的读者可能喜欢让每个使用者去到自己的家目录读取 mail box 的咚咚，亦即是将 /var/spool/mail 的内容给他搬到个别的家目录去！例如 vbird 的 mail box 变成的 /home/vbird/vbird 这个档案！不过，如此一来的话， sendmail 与 pop 都将需要改写其 source code ！所以『不建议这么搞喔！』

本章与 LPI 的关系

在 LPI 网站 http://www.lpi.org 里面提到的，关于 Sendmail 的考试题库的地方，只有在 LPI level 1 的 102 ，里面的 topic 113 Networking Services ，第二点当中，简易的 Sendmail 设定。强调的是『应试者必须简单的设定 sendmail (指的应该是 m4 scripts ，不过会很简单！不要担心～)、能够建立 mail aliases 、能够管理邮件队列、能够启动或者是关闭 sendmail 这个服务、了解使用者的邮件转递 (forward 功能)，以及简单的 sendmail 除错！此外，应试者也需要了解什么是 Open Relay 与避免 Open Relay 才行！』至于会考的档案与指令可能有这些：

- /etc/sendmail.cf (或是 /etc/mail/sendmailcd)
- /etc/aliases 或是 /etc/mail/aliases
- /etc/mail/*
- ~/.forward
- mailq
- sendmail
- newaliases

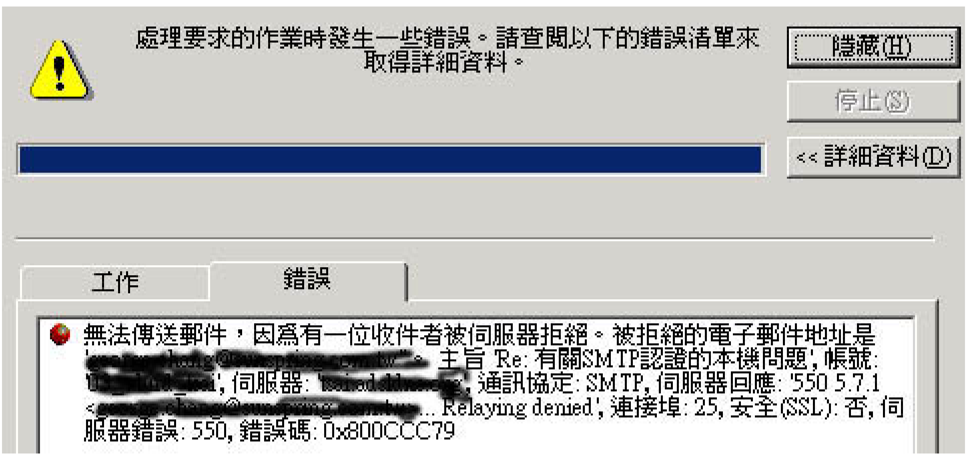
参考资源：

- Sendmail 官方网站： http://www.sendmail.org
- Cyrus-SASL 官方网站： http://asg.web.cmu.edu/cyrus/download/sasl/doc/
- Procmail 官方网站： http://www.procmail.org
- Open Relay Database： http://www.ordb.org/
- Study Area 之邮件架设：http://www.study-area.org/linux/servers/linux_mail.htm
- SMTP 认证系统的建置：http://beta.wsl.sinica.edu.tw/~ylchang/Email/sendmail-auth/
- 台湾学术网络黑名单网页： http://140.111.1.22/tanet/spam.html
- 卧龙小三的 Procmailrc 范例：ftp://ftp.tnc.edu.tw/pub/Sysop/MAIL/procmailrc
- 林克敏主任文件集之 Procmail 范例：http://freebsd.lab.mlc.edu.tw/procmail.htm

---

本章习题练习 (要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 我在使用 Sendmail 寄信时，却发生底下这个问题，请问可能的发生原因为何？



- 请列出四个 Mail Server 的相关的组件，以及其功用为何？
- POP3 与 SMTP 的功能为何？
- 请简单的说明 DNS 里面 MX 标志与 Mail 的关系为何？
- 今天我突然兴起，想要修改我的 sendmail ，请问， sendmail 的设定档在哪里？而我要以什么程序修改 sendmail 呢？
- 什么是 mailling list ？在 sendmail 底下有什么方法可以不藉由其它的软件达到 mailling list 的功能？
- 如何察看邮件队列的内容，以及邮件队列内容放置在何方？
- 若我的 sendmail 主机有很多名称，我想让这些名称都可以进行 mail 的接收，应该修改什么档案？
- 什么是 Open Relay？

简易 Mail Server 架设 -- Postfix 设定

最近更新日期：2004/04/07

在介绍完了 sendmail 之后，您是否觉得～唉！怎么 sendmail.cf 这个设定档这么难搞定～还得藉由 m4 scripts 才能搞定！而且， sendmail 需要考虑到相当多的安全设定项目 (Security)，否则很容易一不小心就被攻击了！那么有没有更简单、更安全的邮件系统啊！当然有啊！那就是 Postfix 这个咚咚！Postfix 的作者有鉴于 sendmail 是这样成功的一个 MTA ，但是偏偏有点小问题，所以该作者站在兼容的立场上面写出这个几乎可以完全取代 sendmail 的 MTA !此外，Postfix 还更加的安全呢，真是很不错的一个 mail server 的替代方案啊！

前言：

在开始介绍 Postfix 这个服务器之前，得先告诉您的是，这个 Postfix 的用途是『邮件服务器』，那么我们在前一章『Sendmail 服务器』里面已经提过了 mail server 的相关原理与运作过程，这个 Postfix 与 sendmail 是类似的东西，那就是『MTA』啦，既然都是 MTA ，使用的协议也相同，同时，这个 Postfix 最早之前的用途也是想要用来『取代 sendmail 』。所以，为了节省笔墨，也为了未来进行修改的时候不要有太多的版本 (意思是这里 copy 一份 mail 原理，那里又有一份原稿～) ，因此上，在进行本章的学习之前，请『务必』前往『简易 Mail server -- sendmail』读一读 Mail server 的原理与相关的说明啊！

本章学习之前你需要先知道的知识：

上面刚刚提到，学习本章你必须要知道 Mail Server 的相关知识才行，在这里，我们不再说明已经提过的咚咚，请自行再前往翻阅。由于架设某种服务器的第一步就是要了解该服务器的工作原理，因此，在架设 Postfix 之前，您至少要知道以下的几个咚咚：

1. Mail Server 能否运作与 DNS (MX 与 A recode)的相关性为何？
2. 什么是 MTA, MUA, MDA 与 Mail box, Mailing list 等相关的术语，及其内容所代表的意义！
3. 什么是 smtp, pop3 以及 imap 协议，他们的用途分别是什么？
4. 什么是 Relay 与 Open Relay ？
5. 什么是 SMTP 邮件认证？
6. 什么是邮件的别名与转递(aliases 与 forward)？
7. 什么是 Procmail 与什么是 ORDB 呢？

如果您不晓得上面问题的答案，请不要『白目』的继续往下看 ^_^，先前往『简易的 sendmail 服务器』瞧一瞧相关的原理之后，再来这里吧！慢点学习不打紧，学的不精....可能会有害啊！ @_@

为什么要有 Postfix 呢？

这是个很有趣的问题：『为什么要有 Postfix 呢？有了 sendmail 不就可以了吗？！』说到这个就要谈到 postfix 的由来了！

Postfix 是由 Wietse Zweitze Venema 先生(http://www.porcupine.org/wietse/)所发展的。早期的 mail server 都是使用 sendmail 架设的，还真的是『仅此一家，绝无分号！』 ^_^ ！不过，Venema 博士觉得 sendmail 虽然很好用，但是毕竟不够安全，尤其效能上面并不十分的理想，最大的困扰是....他的设定档 sendmail.cf 真的是太难懂了!对于网管人员来说，要设定好 sendmail.cf 这个档案，真不是人作的工作～

为了改善这些问题， Venema 博士就在 1998 年利用他老大在 IBM 公司第一个休假年进行一个计划：『设计一个可以取代 sendmail 的软件套件，可以提供网站管理员一个更快速、更安全、而且"完全兼容"于 sendmail 的 mail server 软件！』这个计划还真的成功了！而且也成功的使用在 IBM 内部，可以说是完全取代了 sendmail 这个邮件服务器！在这个计划成功之后， Venema 博士也在 1998 年首次释出这个自行发展的邮件服务器，并定名为 VMailer。不过， IBM 的律师却发现一件事，那就是 VMailer 这个名字与其它已注册的商标很类似，这样可能会引起一些注册上面的困扰。为了避免这个问题，所以 Venema 博士就将名称改为 Postfix ！这个 Postfix 有『在什么什么之后修正』的意思。鸟哥个人认为， Venema 先生最早的构想并不是想要『创造一个全新的 Mail server 软件，而是想要制造一个可以完全兼容于 sendmail 的软件』，所以，Venema 先生认为他自行发展的软件应该是『改良 sendmail 的缺失』，所以才称为 Postfix 吧！取其意为：『在 sendmail 之后的改良的邮件服务器软件！』

所以啦， Postfix 设计的理念上面，主要是针对『想要完全兼容于 sendmail』所设计出来的一款『内在部分完全新颖』的一个邮件服务器软件。就是由于这个理念，因此，Postfix 改善了 sendmail 安全性上面的问题，改良了 mail server 的工作效率，更由于其设定档完全为 ASCII 码，且设定内容都是『人类看的懂得语言！』因此，你可以轻易的由 sendmail 改良到 Postfix 上面！这也是当初 Venema 博士的最初构想啊！就是基于这个构想，所以， Postfix 在外部设定档案的支持度，与 sendmail 几乎没有两样，同样的支持 aliases 这个档案，同样的支持 ~/.forward 这个档案，也同样的支持 SASL 的 SMTP 邮件认证功能等等！所以，呵呵！赶紧来学一学怎样架设 Postfix 这个相当出色的邮件服务器吧！ ^_^

套件安装
跟之前一样的，我们需要的 mail server 功能有哪些呢？

- 具有 smtp 的功能；
- 具有 pop3 的功能；
- 具有 procmail 过滤邮件的功能；
- 具有 Open Relay Data Base 抵挡的功能；

为达成上述的功能，所以你至少需要底下的几个套件：

- cyrus-sasl
- procmail
- postfix
- imap(同时支持 pop3 及 imap 两个协议)

如果您是使用 Mandrake 后期版本的话，那么恭喜您，由于 Mandrake 预设就是使用 Postfix 做为邮件服务器，并且在安装的时候就已经将 Postfix 安装到你的系统当中了。至于 Red Hat 9 同样的也提供了 Postfix 喔！而如果您是使用非 postfix 为邮件服务器的 Linux distribution ，呵呵！仔细的查看后面介绍的 Tarball 安装的方式吧！

使用 RPM 安装完整的 Postfix + POP3 + SMTP + Procmail

底下我们分 Mandrake 9.0 与 Red Hat 9 这两个主要 Linux distribution 来介绍 Postfix + Cyrus-SASL 的方法喔！

Makdrake 9.x 版本
如果你是使用 Mandrake 之类的 Linux distribution 的话，由于他预设是以 Postfix 这个优良的邮件服务器系统，所以您可以不费吹灰之力的，就将 Postfix 以 RPM 安装完毕啰！基本上，如果是 Mandrake 的话，你需要安装的套件大致上有：

- SMTP 认证套件： cyrus-sasl （ cyrus-sasl-1.5.27-5mdk 以及其它认证机制函式库 ）
- Postfix 邮件服务器： postfix （ postfix-1.1.11-4mdk ）
- POP3 服务器： imap （ imap-2001a-9mdk, imap-devel-2001a-9mdk 两个 ）
- 邮件分析软件：procmail （ procmail-3.22-3mdk ）

你至少要安装的套件就有上面这几个，同时，请拿出您的原版光盘将上面的套件全部安装吧！安装的方法我们在『鸟哥的 Linux 私房菜 -- 基础学习篇』介绍过的 RPM 与 Tarball 安装方法里面提过多次了，请自行参考喔！安装完毕之后，你的主机就已经具有 Postfix 这个系统啦！不过，由于我们还要提供 SMTP 以及其它相关的功能，所以这里我们必需要确认一下各个套件是否都完全的安装了呢？底下我们就一个一个的来设定吧！(注：请特别留意 Cyrus-sasl 的版本，因为不同的版本他的函式库所在目录与设定文件都不相同！因此，在本篇文章中，您会发现我使用 Tarball 安装的 2.xx 版本与使用 RPM 安装的 1.5.xx 版本设定档的内容不相同，不要怀疑，确实是如此的喔！ ）

安装 cyrus-sasl

```
1. 安装
# 安装的方法很简单，就是使用 RPM 即可！不过，请先确认是否已经安装！
[root@test root]# rpm -qa | grep sasl
libsasl7-1.5.27-5mdk
cyrus-sasl-1.5.27-5mdk
libsasl7-devel-1.5.27-5mdk
libsasl7-plug-plain-1.5.27-5mdk
libsasl7-plug-login-1.5.27-5mdk

# 如果尚未安装该套件，请拿出你的光盘片， mount 光盘，然后找寻该档案，
# 直接安装他吧！不过要注意的是，上面的档案都要安装喔！因为我们使用的
# 是最基本的 plain 与 login 这两个机制，所以至少我们要安装上面五个咚咚！
[root@test root]# rpm -ivh cyrus-sasl-1.5.27-5mdk.i586.rpm
# 如果有发生属性相依的问题，请自行再将需要的档案自光盘当中找出，
# 然后加以安装吧！

2. 设定与启动
# 由于 Cyrus-sasl 在 Mandrake 当中是 1.5.27 版本，所以他能支持的项目有：
shadow : 使用 /etc/shadow 做为认证码；
pam    : 使用 pam 模块做为认证，这个需要在 /etc/pam.d 这个目录下新增
         一个名为 smtp 的档案，并设定档案内容才行！
sasldb : 使用 SASL 的认证函式库，使用这个功能时，你必需要额外的
         指定 1.函式库的名称； 2.使用 saslpasswd 程序增加使用者！
         鸟哥个人认为这个功能不太好用，因为每新增一个使用者需要主动的
         帮使用者新增账号、密码到认证函式库中，不太方便！
         不过如果您还使用其它的服务器如 LDAP, MySQL 等等软件时，
         则这个功能可就大大的有帮助啦！因为他可以分享账号与密码呐！
pwcheck: 这个功能就不错啦！ pwcheck 是一个服务(daemon)，必需要在
         启动 postfix 之前就启动啦，因为 sasl 的认证就靠这个 daemon，
         基本上，他可以经由读取 /etc/shadow 的资料来提供 client 端
         认证的功能！目前我们 Mandrake 9.0 预设使用的是这个 daemon！
# 给他设定开机时启动 pwcheck 的功能吧！
[root@test root]# vi /etc/rc.d/rc.local
# 在这个档案当中最底下新增一行：
/usr/sbin/pwcheck
# 并且立刻执行这个 daemon 喔：
[root@test root]# /usr/sbin/pwcheck
[root@test root]# ps -aux | grep pwcheck
root   12602  0.0  0.5  1460  348 ?     S  02:59   0:00 /usr/sbin/pwcheck
# 看到没！要出现这个咚咚才算是有启动 pwcheck 喔！

# 在 Sendmail 当中，我们必需建立一个名为 Sendmail.conf 的档案，来告诉
# sasl 我们所需要的认证模式，那么在 postfix 当中也一样，我们必需要建立
```

```
# 一个文件名称为 smtpd.conf 来告诉 sasl ，喝！我 postfix 要的认证方式为何！
# 所以你需要这样做：
[root@test root]# cd /usr/lib/sasl
[root@test sasl]# echo 'pwcheck_method: pwcheck'  >  smtpd.conf

# 上面这样做完之后，系统就知道了：
1. postfix 要用 SMTP 认证时会去读取 /usr/lib/sasl/smtpd.conf 并且知道
   是以 pwcheck 这支程序进行身份认证的动作；
2. 而 pwcheck 这支程序会主动去读取 /etc/shadow ，里面的密码做认证
# 到此为止，就已经设定好了 cyrus sasl 的部分啦！
```

```
安装 postfix
1. 安装
# 安装的方法很简单，就是使用 RPM 即可！不过，请先确认是否已经安装！
[root@test root]# rpm -qa | grep postfix
postfix-1.1.11-4mdk

# 如果尚未安装该套件，请拿出你的光盘片， mount 光盘，然后找寻该档案，
# 直接安装他吧！
[root@test root]# rpm -ivh postfix-1.1.11-4mdk.i586.rpm

# 这样就安装完毕啦！主机设定的部分请参考下一节喔！
```

```
安装 procmail
1. 安装
[root@test root]# rpm -qa | grep procmail
procmail-3.22-3mdk

2. 设定：
# 在设定方面，这主要与 procmail 有关而已，您可以下载 procmailrc
# 并放置到 /etc/ 底下即可详细的数据请参考上一章简易 sendmail 之说明！
```

```
安装 imap
1. 安装
[root@test root]# rpm -qa | grep imap
imap-2001a-9mdk
imap-devel-2001a-9mdk

2. 设定：
# 直接设定成为开机启动即可：
[root@test root]# chkconfig --add ipop3
[root@test root]# /etc/rc.d/init.d/xinetd restart
# 这样就已经可以收信件啦！
```

安装的地方当中，重点在于 Cyrus-SASL 的安装啦！由于我们需要有 SASL 的支持，所以必需要安装这个咚咚！并且也需要设定 smtpd.conf 这个档案！这样就已经差不多 OK 啦！接下来请继续参考主机设定的地方噜！

Red Hat 9 版本：
什么？！Red Hat 9 也同时提供 Postfix 啊？！没错的啦！呵呵！很高兴吧！所以说， Red Hat 9 不但提供了 sendmail 给我们使用，还额外提供了 Postfix 让我们可以随意的转换邮件服务器软件呢！不过，在 Red Hat 9 的 Cyrus-sasl 已经是 2.xx 版本了，而且还主动的提供兼容于 1.5.xx 版本的函式库呢！真是相当的棒啊！但是因为 Red Hat 9 提供的 Postfix 还是使用 1.xx 版，因此我们在 Red Hat 9 预设的 Postfix 邮件服务器中，还是使用 Cyrus SASL 1.5.xx 的函式库喔！所以也就无法使用 saslauthd 这个机制了！并且， Red Hat 9 并没有提供 pwcheck 这个程序，所以在 Red Hat 9 底下的 Cyrus SASL 认证机制就需要使用 sasldb 了！至于安装的方法最简单的方式就是以我们在 网络升级套件 那一章节提到的 APT 来进行安装即可！

```
1. 安装：
[root@test root]# apt-get install postfix
[root@test root]# apt-get install cyrus-sasl
[root@test root]# apt-get install procmail
```

很简单吧！这样就搞定了 Red Hat 9 的 Postfix 啰！(注意， Mandrake 与 Red Hat 不同版本所使用的 Cyrus SASL 机制并不相同喔！在 Mandrake 当中预设是 pwcheck 而在 Red Hat 则预设使用 sasldb 这个验证机制！并且，由于 Red Hat 9 在编译 postfix 的时候使用的是 SASL version 1 ，所以无法使用 saslauthd 这个好用的机制，鸟哥认为，您最好使用 Tarball 来安装您的 Postfix 在您的 Red Hat 9 上面喔！)

使用 Tarball 安装完整的 Postfix + POP3 + SMTP + Procmail (适用任何版本的 Linux 喔！)

以 Tarball 安装 Postfix 也不困难，比起 sendmail 来说，要简单的很多喔！我们这里选择的主要套件有：

- cyrus-sasl-2.1.12.tar.gz
- postfix-2.0.4.tar.gz

至于 procmail 与 imap 两个套件都使用 Linux distribution 提供的 RPM 版本就好了，不需要额外的改装啊！那么底下就来谈一谈怎么安装吧！(注： Cyrus SASL 与 Cyrus SASL2 这两个不同版本的 SASL 函式库放置的目录并不相同，所以你可以分别安装 cyrus version 1 与 version 2 喔！在鸟哥的案例当中，还不会发生问题就是了！另外，如果您是 Red Hat 9 的使用者，由于 Red Hat 9 已经提供了 Cyrus SASL version 2 了，所以您可以直接略过 Cyrus SASL 的安装部分，直接到 Postfix 2.xx 的安装呢！)

```
                    安装 Cyrus SASL 2.xx 版本！
1. 首先将数据解压缩(假设您将我们网站的档案捉到 /root 底下了！)
```

```
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/cyrus-sasl-2.1.12.tar.gz
.....(略).....最后建立一个 cyrus-sasl-2.1.12 的目录

2. 再来就是设定你的 cyrus SASL 的参数以及编译啰!
[root@test src]# cd cyrus-sasl-2.1.12
[root@test cyrus-sasl-2.1.12]# ./configure   \
> --prefix=/usr/local/cyrus-sasl2 --enable-login --enable-plain \
> --enable-pwcheck --with-saslauthd=/var/run \
# 注意上面的语法!!那个 \ 是跳脱字符喔!后面直接 Enter !
# 不要接任何空白或者是 tab 按键!若有问题,请到 Shell 看看!
# 至于那个 pwcheck 的项目,就是要用来做为密码确认的一个咚咚啰!

[root@test cyrus-sasl-2.1.12]# make clean && make
[root@test cyrus-sasl-2.1.12]# make install
# 上面三个步骤会花去一些时间,请耐心等待~
# 而由于我们刚刚设定了 --prefix=/usr/local/cyrus-sasl2 这个参数,
# 所以, make install 之后,我们有用的函式库会在
# /usr/local/cyrus-sasl2/lib/sasl2 这个路径当中,但是, cyrus 程序
# 会主动去找 /usr/lib/sasl2 这个目录!所以,我们必需要做连结档!
# 才可以让未来我们的 postfix 可以直接找的到我们所要的函式库喔!

[root@test cyrus-sasl-2.1.12]# cd /usr/lib
[root@test lib]# ln -s /usr/local/cyrus-sasl2/lib/* .
# 这样就建立起连结档啰!很简单吧! ^_^不过要注意喔,
# 上面这一行最后面那个小数点『.』不要忘记加了!

3. 准备建立 Postfix 与 cyrus SASL 使用的简易设定档:
[root@test lib]# cd /usr/lib/sasl2
[root@test sasl2]# echo 'pwcheck_method: saslauthd' > smtpd.conf
[root@test sasl2]# echo 'mech_list:plain login'  >>  smtpd.conf
# 注意了!一般来说, Postfix 会使用 SASL 这个函式库里面相关的设定档,
# /usr/lib/sasl2/smtpd.conf 这个档案的设定就是 Postfix 的预设使用 SASL 的
# 参数档案,与 1.5.xx 版本不太相同的地方是, 2.xx 版本使用不同的机制:
auxprop  : 使用 sasldb2 这个共享数据库,同样需要使用共享密码档案喔!
          所以一般来说,单纯的 Postfix 比较少使用这种机制;
saslauthd: 使用 saslauthd 这个 daemon 进行认证的工作,所以几乎
          不需要其它的设定值哪,指定 saslauthd 就好啦! ^_^
pwcheck  : 使用与 1.5 版相似的认证 daemon ,不过在 2.xx 版本里面这个模式
          支持度比较没有这么好的啦,所以请爱用 saslauthd 啰!
# 我们使用 SASL 预设的 saslauthd 这支程序做为密码认证的 daemon。
# 至于 mech_list:plain login 是列出支持的认证机制的意思,我们使用的
# 是极为简单的 login 与 plain 两种机制而已!
```

```
4. 建立一些需要的参数：
[root@test sasl2]# vi /etc/man.config
# 新增底下这一行之后，未来我们就可以透过 man 这个工具来查询 sasl
# 相关的指令的用法了！而不需要修改任何咚咚！不过要注意的是，
# 这个档案在每个 Linux distributions 当中不见得相同，例如 Open Linux
# 檔名是 /etc/man.conf 呢！
MANPATH /usr/local/cyrus-sasl2/man

5. 检验 saslauthd 这支程序是否可行！
# 在 cyrus-sasl 的原始码里面提供了一支小程序用来判断 saslauthd 的认证机制
# 是否成功的启动了，这个小程序就是 testsaslauthd 啰！在刚刚原始码目录下，
# 所以你可以这样做：
[root@test sasl2]# /usr/local/cyrus-sasl2/sbin/saslauthd -a shadow
# 执行之后， saslauthd 的 PID 会被纪录到 /var/run/mux.pid 这个档案！
[root@test sasl2]# cd /usr/local/src/cyrus-sasl-2.1.12/saslauthd/
[root@test saslauthd]# make testsaslauthd
[root@test saslauthd]# ./testsaslauthd -u userID -p 'yours.passwd'
0: OK "Success."
# 若显示 OK 的话！那么就是成功啦！很好！我喜欢～

6. 设定开机时启动
[root @test saslauthd]# vi /etc/rc.d/rc.local
# 加入这一行：
/usr/local/cyrus-sasl2/sbin/saslauthd -a shadow
```

安装 Postfix 2.xx 版本！

```
0. 先确认有没有这个档案存在！因为我们的 Postfix 会使用到很多数据库啊！
[root@test root]# locate pcre.h | grep include
/usr/include/pcre.h
# 这个档案也有可能存在于 /usr/include/pcre/pcre.h 里面！不要担心，存在就好了
# 如果没有存在的话，例如 Mandrake 9.0 预设状态可能并不会主动安装这个套件，
# 请自行拿出原版光盘，安装 MDK 的 libpcre0xxxx 档案，至于 redhat 等
# 其它版本，请自行安装 pcre 相关的套件吧！在 Red Hat 9 则是需要
# pcre-devel 这个套件喔！ ( apt-get install pcre-devel)

1. 首先将数据解压缩(假设您将我们网站的档案捉到 /root 底下了！)
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/postfix-2.0.4.tar.gz
.....(略).....最后建立一个 postfix-2.0.4 的目录

2. 清除规则并且建立新的使用 SMTP 的规则给 postfix
[root@test src]# cd /usr/local/src/postfix-2.0.4
[root@test postfix-2.0.4]# make tidy  # 清除规则
[root@test postfix-2.0.4]# make makefiles CCARGS="-DUSE_SASL_AUTH \
```

```
>  -I/usr/local/cyrus-sasl2/include/sasl/"  \
>  AUXLIBS="-L/usr/local/cyrus-sasl2/lib -lsasl2"
# 上面的动作在建立 Makefile 啰！其中那个 -CCARGS 后面接的 -DUSE_SASL_AUHT
# -I/usr/local/cyrus-sasl2/include/sasl 其中 -I 后面的路径就是 SASL2 的
# 函式库所在的目录喔！请依照您安装的 sasl2 目录所在而定！
[root@test postfix-2.0.4]# make
# 就是开始编译啦！过程有点久喔！如果有出现任何 Error 时，
# 请将 error 仔细的查看一下吧！通常最大的原因都是一些
# include 档案没有安装，也就是某些重要的套件没有安装之故，
# 例如 pcre.h 这个档案就是一个例子啰！
# 解决的方法就是将该缺乏的套件安装进去系统啦！
# 如果是 Red Hat 9 的话，可以这样下达指令：
# make makefiles CCARGS="-DUSE_SASL_AUTH -I/usr/include/sasl" \
# AUXLIBS="-L/usr/lib/sasl2 -lsasl2"
# 因为 Red Hat 9 的 SASL2 路径在 /usr/include/sasl ，这里要特别的强调，
# 否则由于 Red Hed 9 同时提供 Cyrus SASL 1.5.x 以及 2.x.x 的版本，
# 可能会造成程序的误判，那么很可能会出现下列的错误喔：
# fatal: SASL per-connection security setup
# 上面的错误讯息出现在 /var/log/maillog 中！

3. 安装前准备工作：
# 如果您的系统是由 sendmail 要改换到 Postfix 的话，你不需要移除 sendmail ，
# 不过却需要进行一些小手术喔！
[root@test postfix-2.0.4]# mv /usr/sbin/sendmail /usr/sbin/sendmail.OFF
[root@test postfix-2.0.4]# mv /usr/bin/newaliases /usr/bin/newaliases.OFF
[root@test postfix-2.0.4]# mv /usr/bin/mailq /usr/bin/mailq.OFF
[root@test postfix-2.0.4]# chmod 755 /usr/sbin/sendmail.OFF \
>  /usr/bin/newaliases.OFF /usr/bin/mailq.OFF

# 此外，还需要建立一个名为 postdrop 的群组与 postfix 的使用者喔！
[root@test postfix-2.0.4]# groupadd -r postdrop
[root@test postfix-2.0.4]# useradd -r -s /bin/false postfix
# 建立一个系统数据账号呐！
[root@test postfix-2.0.4]# echo 'postfix:  root' >> /etc/aliases
# 请注意，如果您之前安装过 sendmail 的话，那么不同版本的 sendmail 他的
aliases 放置的目录并不相同，所以上面这行指令请依您的主机来设定！

4. 开始安装 Postfix 到 /etc/postfix 这个目录下：
[root@test postfix-2.0.4]# make install
# 底下会出现一大堆的问题集，不管如何，几乎按下 enter 就对啦！
# 除了底下这两个问题(最后两个)可以选择将 readme 档案存下来喔！
Please specify the destination directory for the Postfix sample
configuration files.
```

```
sample_directory: [/etc/postfix] /etc/postfix/sample

Please specify the destination directory for the Postfix README
files. Specify "no" if you do not want to install these files.
readme_directory: [no] /etc/postfix/readme_files

# 这样就安装成功啦！接下来请开始查阅 主机的设定 那一节啰！
```

主机的设定

安装完毕 postfix 之后，接着下来就是设定了！同样的，在设定主机之前，有几个东西是您必需先取得的喔：

1. 具有合法的 hostname ，或者是经过授权的 DNS 主机来设定您自己的主机名称(hostname)；
2. 您的 hostname 最好拥有一个 MX 的纪录(record)；
3. 至少一定要了解到什么是 Relay 的问题，以及规划一下您所想要开放 relay 的网域；

你至少需要有这些数据才行呐！好了，我先假设一下我的案例好了：

4. 我的 hostname 为 vbird.adsldns.org；
5. 我的 hostname 还有个别名为 www.vbird.adsldns.org ，也就是说，vbird.adsldns.org 与 www.vbird.adsldns.org 都指向同一个 IP ；
6. 我的 domain name 为 adsldns.org ；
7. 我预计开放的 relay 网域为 vbird.org 这个网域，与内部的 192.168.1.0/24 这个网域，还要可以使用 access 这个档案的设定功能！

开始来了解 postfix 啰！ ^_^

Postfix 的结构

在主机的设定之前我们得先了解一下 postfix 的整体构造，以方便以后来处理我们的档案呐！所以底下我们会先针对 postfix 这个套件的结构做个简单的说明，然后再针对各个设定项目来进行说明啰!Postfix 的设定档几乎完全都在 /etc/postfix 里面,至于执行档则在 /usr/sbin 里面,我们分别来谈一谈几个主要的注意事项吧！

- 设定档：Postfix 的设定档都在 /etc/postfix 里面，主要的设定档有：

        - /etc/postfix/main.cf ：这个就是最重要的 postfix 的设定档了！等一下我们谈到的设定都在这个档案里面进行修改的动作！基本上，他本身就是一个简单的说明文件档啦！不过，要注意的是，在你修改完成了 main.cf 之后，请记得『一定要重新 reload 或重新 restart postfix 才行！』

        - /etc/postfix/master.cf：这个档案是另外一个重要的 postfix 设定档！他主要是规定了 postfix 每个程序的运作参数！

        - /etc/postfix/access ：这个档案与 sendmail 的 access 是相同的！都是用来做为 relay 或者是 deny 某些 IP 与 hostname 的档案！不过，要启用他还是得要修改 main.cf 才行！

        - /etc/postfix/aliases ： 这个档案与 sendmail 的 aliases 也是相同的！同样的可以做为别名之用，所以您可以回头去翻一翻这个档案的用法喔！

        - /etc/postfix/pcre_table regexp_table relocated ： 这几个档案可以做为邮件的过滤之用喔！可以使用正规表示法来进行邮件过滤(filter)的规则，呵呵！会使用这几个档案，或许连 procmail 也不需要使用了呢！ ^_^

    - 执行档：Postfix 的执行档可不少啊！约略提几个主要的执行档吧！

        - /usr/sbin/postfix：这就是 postfix 的主要执行档案啦！启动与简单的关闭 postfix 可以使用：

          ```
          postfix check ：检查 postfix 相关的档案、权限等是否正确！
          postfix start ：开始 postfix 的执行
          postfix stop  ：关闭 postfix
          postfix flush ：强制将目前正在邮件队列的邮件寄出！
          postfix reload：重新读入设定档，也就是 /etc/postfix/main.cf
          ```

          要注意的是，每次更动了 /etc/postfix/main.cf 之后，一定需要执行 postfix reload 喔！

        - /usr/sbin/postalias：这个指令是 sendmail 的 newaliases 啦！他可以用来将上面提到的设定档 aliases 制作成为 postfix 看的懂得『数据库』格式化

档案！由于一般来说，我们都是使用 hash 这一种数据格式，所以啰，你可以简单的使用底下的指令方式来格式化。

```
postalias 数据库格式:檔名
postalias hash:/etc/postfix/aliases  ==>自动建立
/etc/postfix/aliases.db 这个数据库档案！
```

- /usr/sbin/postcat ：这个指令可以用来观察某个邮件队列里面档案的信息！postfix 的邮件队列放置在 /var/spool/postfix 里面，但是这个目录里面的档案格式是 postfix 看的懂得，我们人类看不懂～为了取得这些在邮件队列里面的档案信息，所以我们得使用 postcat 这个指令来读取出信息喔！假如有个档案放在 /var/spool/postfix/deferred，你可以这样看：

  ```
  postcat /var/spool/postfix/deferred/filename
  ```

- /usr/sbin/postconf ：可以用来读取 main.cf 这个档案里面的设定数据的一个指令！用途多多啊！简单的来说，可以直接将你的 main.cf 里面的设定分门别类的显示给你看，可以帮助除错 (debug) 啦！不错的工具，鸟哥个人蛮喜欢加入 -n 这个参数来了解目前的主要规范文件放置在那个目录下！

  ```
  postconf -n
  ```

  显示的结果为：

  ```
  alias_database = hash:/etc/postfix/aliases
  alias_maps = hash:/etc/postfix/aliases
  command_directory = /usr/sbin
  config_directory = /etc/postfix
  daemon_directory = /usr/lib/postfix
  debug_peer_level = 2
  .......
  ```

  看到了吧！你想要知道的数据立刻一目了然，可以加快你的除错喔！不必每次都要进入 main.cf 里面找好久呢！

o

- /usr/sbin/postmap：怪怪！这个指令的用法跟上面提到的 postalias 几乎完全一模一样啊！ ^_^不过，他是用来取代 sendmail 的 makemap 的！用法是：

  ```
  postmap hash:/etc/postfix/access ==>自动建立
  /etc/postfix/access.db 这个数据库档案！
  ```

o

        - /usr/sbin/postqueue ：这个是在观察邮件队列的一个指令啦！你可以使用 postqueue -p 来显示目前的邮件队列内容！这个：

          postqueue -p 与 mailq 显示的内容会一模一样喔！

    - 目录：同样的具有邮件队列与邮件信箱的放置目录啊！
        - /var/spool/mail ：电子邮件信箱放置目录
        - /var/spool/postfix ：邮件队列，里面还有很多次目录，都是一些其它功能的咚咚啊！

整体的架构大致是如此，好啦！了解的一定程度的架构之后，我们要开始来谈谈主机的最基础设定了吧！

基础设定

终于要来搞定我们的设定档 /etc/postfix/main.cf 啦！先来搞定比较简单的几个部分喔！在 Postfix 安装完毕之初，他仅支持本机寄信而已，如果您想要开放外送邮件，那就必需要做一些手术啦！底下几个重要的项目先来观察一下啰！(注：请使用 vi 来编辑喔！)

在 main.cf 这个档案当中，设定的项目都是以类似变量的设定方法来设定的，例如要设定 postfix 主机名称时，就以： myhostname = vbird.adsldns.org 这样的方式来设定的，那么什么是『变量』呢？如果还不清楚，赶紧拿出『鸟哥的 Linux 私房菜 -- 基础学习篇』翻到『认识 BASH 』的章节，好好的瞧一瞧去吧！尤其变量的内容显示需要用 $ 来领头的用法，千万不可忘记！例如： myorigin = $myhostname ==> myorigin = vbird.adsldns.org 上面两个是一模一样的，只是藉由变量来让我们的设定更加的简单化！不过，与变量设定规则当中比较不同的，在 main.cf 当中的等号两边需要有空白！此外，如果有两个以上的设定呢？那你就必须要逗号『,』或者是空格符『』来做为分隔了！例如： mydestination = $myhostname, $mydomain mydestination = $myhostname $mydomain 在底下的设定当中请特别留意这些设定喔！

    - 主机名称的设定：myhostname 与 mydomain (极重要)
      这个设定蛮重要的喔！就是在设定你的主机名称啰！请特别留意，这个『 myhostname 』的变量在 main.cf 设定档的后面会持续的被使用到，所以，不要写错了！至于 mydomain 则是你的网域啦！以鸟哥的 linux.vbird.org 为例，我的主机名称为 linux.vbird.org (用 FQDN 的方式来写)，此时，我的 domain 就是 vbird.org 啦！在本章的例子当中，主机名称为 vbird.adsldns.org 网域名称为 adsldns.org ，所以你要这样设定：

```
myhostname = vbird.adsldns.org    <==这里请书写你的主机名称喔!
mydomain = adsldns.org            <==这里则是你的 domain 名称!
```

-
- 送件来源的主机名称：myorigin
  这个项目在设定『邮件标头上面的 mail from 的那个地址』！当我们在本机端使用 mail 这个程序发送信件时，由于没有定义送件者的地址，一般而言， 邮件主机会以目前的主机名称做为邮件的 FROM 的那个主机名称。例如在我的 postfix 上面使用 mail 发送信件时，在发送的邮件当中，就会显示发信人为 vbird@vbird.adsldns.org 的意思啦！不过，我也可以自行指定来源主机的名称，就是在这个 myorigin 设定的！通常，使用 $myhostname 来设定 myorigin 即可，不过，在某些大型主机当中，由于这个大型网域内可能有多部的邮件主机，在此时可指定 myorigin 为 $mydomain ，不过，如此一来还得替每个 user 指定其它相关的参数！所以，如果没有其它要求的话，这个地方设定为底下的样子即可：

```
myorigin = $myhostname
```

-
- 可用来收件的主机名称：mydestination (极重要)
  这个项目可就重要的太多太多了！这个项目就是 sendmail 当中的 /etc/mail/local-host-names 相同的意义啦！你的 postfix 只有在发现 mydestination 这个项目中有写入的主机名称做为收信主机时，才会将该封信件收下来！例如，当你的主机名称有 vbird.adsldns.org 与 www.vbird.adsldns.org ，这两个主机名称均指向同一部主机，那么只有在你将这两个名称都写入 mydestination 项目中，那么两个主机的来信才可以被接受，否则就会被退回喔！这个项目有多种设定方法，最简单的就是直接写入档案中，另外，也可以设定的跟 sendmail 一样，用外部的档案来取代喔！

```
方法一：利用变量的型态
mydestination = $myhostname, www.$myhostname

方法二：直接给他写入名称
mydestination = vbird.adsldns.org, www.vbird.adsldns.org

方法三：使用档案型态
mydestination = /etc/postfix/local-host-names
上面的名称可以随便你取喔！然后在该档案内设定
[root@test root]# vi /etc/postfix/local-host-names
vbird.adsldns.org,
www.vbird.adsldns.org
```

- 一般来说，除非您对于 local-host-names 这个文件名称情有独锺，否则的话，鸟哥个人建议您直接在 mail.cf 里面直接设定好你的可接受的主机名称即可！特别留意的是，如果你的 DNS 里头的设定有 MX 的话，那么请将 MX 指向的那个主机名称一定要写在这

个 mydestination 内！否则很容易出现错误讯息喔！特别的给他注意这一点！『一般来说，使用者最常发生错误的地方就在这个设定里头呢！需要搭配你的 DNS 设定喔！』

- 简易的 Relay 控制：inet_interfaces mynetworks_style mynetworks relay_domains (极重要)
  这个项目在控制谁可以利用我们的主机来寄信呢？也就是在 sendmail 里面的 access 那个档案的咚咚啦！分别说明一下几个东西啰！

  - inet_interfaces：你的 Postfix 主机能被用来使用的接口，假如你的 Linux 主机有多个接口，例如多张网络卡或者是拨接之后又会产生的 ppp0 这些接口时，如果你不想要全部的接口都开放 postfix 功能，那么就可以在这里指定能用的接口啰（注：指定的方式是以主机名称为主喔！）一般预设只有自己的内部循环网络可以使用（localhost），不过，如果要连上 Internet 的话，建议就要全部都开放啦！ inet_interfaces = all

  - mynetworks_style:这是用来设定你所想要的 relay 的信任(trust)网域型态！一般来说，有三种主要的型态，分别是：

    - class ：表示为 A/B/C 三种 class 其中之一，在拨接或者是 ADSL 的情况下，这种型态设定并不好！因为他会主动的去找你目前的 IP 所在的网域来进行 relay 的功能开放！举个例子来说，如果你是以中华电信的 ADSL 拨接情况，那么你的 IP 很可能会是 61.59.xxx.yyy ，那如果你设定为 A class 的话（ postfix 会自动的判断，请至"网络基础"一文当中查看 A/B/C 三个 class 的说明 ），那么只要是 61 开头的 IP 都可以用你的 postfix 喔！很严重吧！这个设定通常只给内部私有网域来使用的啰！

    - subnet：这是 postfix 的默认值，使用 subnet 的型态来设定喔！意义是说，你的网络卡 IP 所在网域的任何一个 IP 都会被接受的意思！例如我的主机网卡私有 IP 为 192.168.1.2 那么所有我内部网域连接到这个网卡的 192.168.1.0/24 这个网域的所有 IP 都会被认为是『合法的』！而自动的提供其 Relay 的功能呢！

    - host ：在这个设定时， postfix 仅会知道 localhost 设定为 trust (信任) 的网域而已！

基本上，这个设定值你可以指定为 mynetworks_style = subnet ，不过，也可以不要设定啦！直接以底下 mynetworks 来设定 relay 的网域即可！另外，如果您有设定 mynetworks_style 以及 mynetwork 时，那么 mynetworks 这个设定会取代掉 mynetworks_style 喔！因为如此，所以鸟哥个人是不设定 mynetworks_style 的，只设定 mynetworks 而已！

- mynetworks ：这个也是用来开放 Relay domain 的一个设定项目！一般来说，也可以设定成很多的方式，包括档案与变量或直接书写需要的 IP/netmask 类型！这里我们假设有 192.168.1.0/24, 127.0.0.0/8 以及 192.168.1.100 这几个咚咚要开放，所以我可以这样写：『mynetworks = 192.168.1.0/24, 127.0.0.0/8, 192.168.1.100/32』！如此一来，连 access 都不需要设定呢！^_^ (注：请注意，如果你没有设定 mynetworks 的话，一定要将 mynetworks_style 设定为 host 喔!不然你的 IP 所在的子网域的 IP 会被自动的认为是『合法的』呢！)另外，如果你想要启用 /etc/postfix/access 这个档案的设定功能时，那就必需要再加以修改喔！例如：『mynetworks = 127.0.0.0/8, hash:/etc/postfix/access 』注意文件名称使用完整档名！

- relay_domains：相对于 mynetworks 设定的专门针对『来源』的 IP 来设定，那么如果是以主机名称，或者是领域名称(domain)时，要如何设定呢？那就可以使用这个项目来设定啦！ mynetworks 设定『信任网域的来源 IP 』而 relay_domains 则可以设定『信任网域的来源与目标之主机或领域名称』啦！举个例子来说，如果你的主机要开放 vbird.org 这个网域的主机的 Relay 功能，那么你将 vbird.org 写入 relay_domains 当中时，那么：

  1. 任何由 vbird.org 来的信件都会被认为是『信任』的，所以 postfix 主机会自动帮忙 relay ；
  2. 由任何地方来的信件，"并且"要往 vbird.org 这个网域去的信件，postfix 主机也会帮忙 Relay 的！

  所以说，这个 relay_domains 可以设定『来源』与『目标』的主机或领域名称喔！一般来说，relay_domains 预设就是我们自己的主机啦！！

上面的设定项目当中，我们可以仅设定两个即可啦！其它的不用管也没有关系！不过，由于我预设还有启动 vbird.org 这个 relay domain ，所以说，整体架设如下：

```
inet_interfaces = all
mynetworks = 192.168.1.0/24, 127.0.0.0/8, 192.168.1.100/32,
```

```
      hash:/etc/postfix/access
relay_domains = vbird.org

设定完成之后还需要启动 access 喔!
1. 先手动修改完任何你想要的动作在 /etc/postfix/access 当中;
2. 完成数据库的建置:
[root@test root]# postmap hash:/etc/postfix/access
```

- 设定账号别名的数据库：aliases_maps (极重要)
  还记得 aliases 这个账号别名的用途吧？！如果你是以 RPM 的方式来安装 Postfix ，那么这个设定值是没有问题的，但是，如果你是以 Tarball 来安装你的系统时，并且是由 sendmail 升级的，那么由于 sendmail 将 aliases 放置在 /etc/aliases (要视你的 Linux 版本而定！)，不过，比较好的作法是将 aliases 放置在 /etc/postfix/aliases 里面，这样设定与目录之间的关系比较容易找啦！那要如何修改呢？首先，你可以将旧有的 aliases 移动到 /etc/postfix/aliases 里面，并且请特别注意，这个旧有的档案里面，一定要存在有『 postfix: root 』这一个设定才行喔！然后透过 /etc/postfix/main.cf 里面的 alias_maps 来修改，通常鸟哥个人喜欢改成这样：

```
alias_maps = hash:/etc/postfix/aliases
alias_database = hash:/etc/postfix/aliases
```

- 改完之后还需要以 postalias 来建置成为 postfix 可以读取的数据库格式喔！

```
[root@test root]# postalias hash:/etc/postfix/aliases
```

-

呵呵！设定完上面这几个重要的项目之后，基本上，你的 postfix 主机『已经准备好上路啦！』所以，这个时候请将他启动吧！或者是重新启动他吧！

```
先确认设定档有没有错误:
[root@test root]# postfix check  <==检查错误，若没有任何显示，表示正确

如果是 RPM 安装的情况:
[root@test root]# /etc/rc.d/init.d/postfix restart

如果是 Tarball 安装的情况:
[root@test root]# postfix start

检查是否正确的启动了呢?
```

```
[root@test root]# netstat -tl | grep smtp
tcp        0      0 *:smtp                  *:*                         LISTEN

设定开机的时候立刻启动:
[root@test root]# chkconfig --add postfix <== RPM 安装时
[root@test root]# vi /etc/rc.d/rc.local    <== Tarball 安装时
加入下面这行:
/usr/sbin/postfix start
```

当出现上面的那行斜体字显示的内容 (LISTEN) 时，哈哈！你的 postfix 已经启动啦！这个时候 postfix 已经可以：

23. 针对 mynetworks 设定的 Client 端，进行 relay 的动作；
24. 针对信件目的为 mydestination 设定的主机名称『接收该信件』！

虽然你是使用 Tarball 方式安装 Postfix 的，但是如果你想要使用 /etc/rc.d/init.d/postfix restart 的语法来启动的话，鸟哥已经写了一支简单的 scripts 提供大家这样进行工作呢！可以前往下载(http://linux.vbird.org/download/index.php#sendmail_postfix) 喔！

重要观念：Postfix 预设的 Relay 流程与收受信件流程：

在预设的情况下 (也就是说，在没有开放 SMTP 这个邮件认证时！) Postfix 对于收信与寄信的流程是如何呢？！这里我们得分别来谈一谈几个主要的设定项目与邮件传输的动作喔！

- 帮助 Client 发送信件，也就是 Relay 的功能开放的需求项目：在开放哪些项目之后，你的 postfix 主机才会帮助『信任』的 Client 端 Relay 呢？
  1. 当 Client 来自信任的网域，也就是 IP 符合 $mynetworks 的设定值时；
  2. 当 Client 来自信任的机器，也就是主机名称符合 $relay_domains 的设定项目时；
  3. 当 Client 来自不信任的网域，但是去的目的地主机端符合 $relay_domains 的设定时。

  当符合上面三点的任何一点时，那么 postfix 将对该信件进行 relay 的动作喔！

- 收信程序：在哪些情况之下，我们的 postfix 会将该封信件收下来呢？
  0. 收件者主机名称符合 $inet_interfaces 的设定；
  1. 收件者主机名称符合 $mydestination 的设定；
  2. 收件者主机名称符合 $virtual_maps 的设定。

符合上面三点的任何一项时，那么该封信件就会被我们的 postfix 收下来！所以，如果你的 postfix 能寄不能收，或者是能收不能寄，请仔细的观察一下上面的几个重大的设定项目，仔细的在你的 main.cf 重新设定一下，基本上，应该就不会有太大的问题啦！

启动 smtp 邮件认证功能

谈完了 Relay 的功能之后，接下来自然就是重要的『SMTP』邮件认证的部分了！目前邮件认证的功能有相当多种类，请特别留意的是，鸟哥这里使用的是 SASL 释出的 pwcheck 与 saslauthd 这两个 daemons 提供的功能喔！这两个功能分别在 cyrus-sasl 1.5.xx(pwcheck) 里头与 2.xx(saslauthd) 里头设定的啦！但是 Red Hat 9 使用的是 saslauthd 喔！请回到前面的 Postfix 安装里面瞧一瞧 RPM 的 1.5.xx 与 Tarball 的 2.xx 好吗！你要作的动作有：

27. 确定 cyrus-sasl 已经安装，并且同时必须要安装 libsasl 相关的 LOGIN 及 PLAIN 的函式库(这个在前面 Postfix 套件安装 当中已经说明了，分别参考 Tarball 与 RPM 的安装，请前往参考)；
28. 确定在 sasl 函式库内已经存在有 smtpd.conf 这个档案 (注意，cyrus-sasl 第一版函式库在 /usr/lib/sasl 而第二版则在 /usr/lib/sasl2 ，目录并不相同，请仔细察看一下您的 sasl 喔！)，这个档案的内容在 Postfix 套件安装里面已经说明了！并且 sasl 与 sasl2 设定并不相同，请自行参考；
29. 确定已经使用 pwcheck(cyrus-sasl 1.5.xx) 或 saslauthd(cyrus-sasl 2.xx) 这两个 daemon 了，并且已经成功的启动 daemon 了！可以使用『ps -aux | grep pwcheck 』或『ps -aux | grep saslauthd 』察看是否有 process 存在；
30. 修改 /etc/postfix/main.cf 这个档案的设定 (请参考底下的说明)；
31. 重新启动 postfix 。

基本上，大部分的内容我们在安装的时候已经搞定啦，就剩下需要设定 main.cf 这个档案而已，那么有哪些数据必须要设定的呢？

针对主机设定：

- smtpd_sasl_auth_enable：『确定是否要针对 Client 启动 sasl 的认证呢？』预设是不启用，这里我们必须要将他给启用才行啊！所以要『smtpd_sasl_auth_enable = yes』

- smtpd_sasl_local_domain：『确认已经经过认证的网域』，就是不需要身份认证也可以是『信任』的网域啰，在 1.5.xx 版的 cyrus 可以直接填写为 $myhostname 啦！不过，在 2.xx 版本时，就不能设定了！很重要喔！

- smtpd_recipient_restrictions：『信件收件的限制规则』，既然已经启动了 sasl 的邮件认证，此时必须要规定一下，到底在什么条件之下，我们的 postfix 可以接受 Client 端的 Relay 的功能呢？主要有底下这几种限制的规则：
    - permit_mynetworks：在 mynetworks 这个项目设定的网域 IP 都可以被允许联机喔；
    - permit_authenticated：允许使用者经过 SASL 的认证方式寄信！
    - check_relay_domains：通过一些测试之后的主机可以进行 relay ，与刚刚前几节的 relay_domains 有点关系！不过，在 2.xx 版本下，需要变成底下的项目了！
    - reject_unauth_destination：这是在 2.xx 版本时的设定项目，取代了 check_relay_domains 的设定项目啰！

  通常我们会设定上面这三个就差不多啦！

- smtpd_client_restrictions：『针对 client 端的限制规则』，经过 mynetworks 这个信任网域的 IP 之后，那未经信任的 Client 端 IP 来源你要限制他使用你的 postfix 时，这个项目就要启动 permit_sasl_authenticated 啰！

- smtpd_sasl_security_options：『限制某些登入的方式』，在 Postfix 里面，预设是使用 Plaintext 的方式来认证的，所以自然不能取消掉这个认证啰(noplaintext)！但是我们可以取消掉匿名登入的型态喔(noanonymous)！可以这样做『smtpd_sasl_security_options = noanonymous』

针对 Client 设定：

- smtp_sasl_auth_enable：上面是针对主机来设定的，这个项目则是针对 Client 来设定的，在预设的情况之下， Postfix 并不会对 client 提供认证的功能，也就是说，postfix 只会依据 mynetworks 之类的信任网域来提供 Relay 的功能，那么我们要启动对于 client 的身份认证功能时，这里就必需要设定为 yes 才行！不过，如果是在 2.xx 版本时，这个设定就不需要啦！

所以整个设定值就变成这样啦：

```
[root@test root]# vi /etc/postfix/main.cf
1.5.xx 版本的 Cyrus SASL
底下请自行新增在这个档案的最后面：
smtpd_sasl_auth_enable = yes
smtpd_sasl_local_domain = $myhostname
smtpd_recipient_restrictions = permit_mynetworks, permit_sasl_authenticated,
```

```
    check_relay_domains
smtpd_client_restrictions = permit_sasl_authenticated
smtpd_sasl_security_options = noanonymous
smtp_sasl_auth_enable = yes

2.xx 版本的 Cyrus SASL
底下请自行新增在这个档案的最后面：
smtpd_sasl_auth_enable = yes
smtpd_sasl_local_domain = ' '
smtpd_recipient_restrictions = permit_mynetworks, permit_sasl_authenticated,
     reject_unauth_destination
smtpd_client_restrictions = permit_sasl_authenticated
smtpd_sasl_security_options = noanonymous
注：有网友来信告知，在 smtpd_sasl_local_domain 的部分，也可以修改成：
smtpd_sasl_local_domain=
即可！
```

另外，由于 RPM 版本的 Postfix 预设会使用 chroot jail 这个比较安全的动作，所以，如果是 RPM 安装时，你还必需要修改一个档案：

```
[root@test root]# vi /etc/postfix/master.cf
找到底下的这一行：
smtp    inet    n       -       y       -       -       smtpd
将他改成为：
smtp    inet    n       -       n       -       -       smtpd
```

注：chroot jail 的功能在于『使用权限较低的一般身份使用者来进行 postfix 的工作程序，只有在需要的时候才可以进入 /var/spool/postfix 这个邮件队列目录！』，对于系统来说，是有一定程度的安全保障的！所以， Postfix 才会在预设的情况之下以 chroot 的功能来进行 postfix 的！不过，再加上了 SMTP 这个认证机制之后，由于他必需要以比较高等级的使用者来执行一些认证个功能，所以除非您额外的加入很多的函式库去到 chroot jail 目录下，否则的话，就不能使用 chroot 啦！所以，如果要启动 SMTP 的话，请务必将 master.cf 这个档案修改过喔！^_^

这样就算设定完毕啦！然后重新启动 postfix 看看情况是如何：

```
[root@test root]# /etc/rc.d/init.d/postfix restart
[root@test root]# telnet localhost 25
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
220 vbird.adsldns.org ESMTP Postfix (1.1.11) (Mandrake Linux)
ehlo localhost   <==确认一下你的主机状态
```

```
250-vbird.adsldns.org
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-AUTH PLAIN LOGIN  <==出现这个就是成功啰!
250-XVERP
250 8BITMIME
quit  <==离开吧!
221 Bye
Connection closed by foreign host.
```

然后确定一下你的机器是否真的有在进行认证的工作呢？首先，先取得你的密码，然后再以密码来测试看看认证是否可以通过！举例来说，假如我有一个使用者 test ，test 的密码为 abc 的时候，那么你可以这样做：

```
[root@test root]# printf 'test\0test\0abc' | mmencode
dGVzdAB0ZXN0AGFiYw==   <==这个东西就是你的密码啦!
[root @test root]# telnet localhost 25
telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 vbird.adsldns.org ESMTP Postfix
ehlo localhost   <==先打招呼
250-vbird.adsldns.org
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-AUTH LOGIN PLAIN
250-XVERP
250 8BITMIME
auth plain dGVzdAB0ZXN0AGFiYw==
235 Authentication successful  <==若出现这一行表示你的 SMTP 已经运作正常
quit
```

几个相关的档案说明

在 sendmail 当中有几个颇为重要的档案，例如 ~/.forward 等等的档案！在这里我们也必需说明一下这几个档案在 postfix 里面的应用情况：

- /etc/mail/local-host-names：这个档案是用来书写你的 mail server 可以接收『目的 e-mail server』的名称的一个档案，这在先前的设定项目中已经提过了，就是 /etc/postfix/main.cf 里面的 mydestination 的设定内容啦！

- /etc/mail/access ：这个档案用来规定可以 Relay 或者需要 discard 的动作！在 postfix 当中预设是不开启这个档案的设定的，如果要启用的话，需要在 mynetworks 里面规定好文件名称喔！设定方法请参考前几节的说明！

- ~/.forward ：这个档案在 sendmail 与 postfix 当中的设定是一模一样的，都是用来帮忙邮件的转递的，可以参考 sendmail 的设定项目！

- /usr/sbin/mailq：这个执行档已经被更改过了！目前可以使用 postqueue -p 来显示出还在邮件队列的信件标题喔！

基本上，在 sendmail 当中可以使用的外部设定档案，在 postfix 底下几乎都能再被使用！所以，由 sendmail 升级到 postfix 真是相当的简便啊！ ^_^

客户端的使用说明

所有在 sendmail 当中需要注意的与可以使用的 Client 端的功能，在 postfix 当中都同样的可以使用喔！所以，这部份请回到『sendmail 服务器』那一章去瞧一瞧怎么使用 client 来进行 Mail server 的种种工作呐！其中，需要特别强调的有几个小细节：

- 如果您有启动 SMTP 的认证时，请千万注意在 client 端的 MUA 必需要启动『我的寄件人需要密码』的项目，详情请参考 sendmail 的设定一文；
- 在 sendmail 当中，不论 port 25 有没有启动， Linux 本机上面使用 mail 寄信，『还是可以』将信传送出去，不过在 postfix 可不是这么回事了！你若没有启动 port 25 时，那么使用 mail 将会把信暂时的放置邮件队列当中 (/var/spool/postfix) ，直到再次开启 port 25 之后，信件才有可能再次的备传送出去喔！

关于邮件主机安全的设定

关于邮件主机的安全性方面，我们已经启动了 SMTP 了，那么还有 Open Relay 主机的抵挡，以及 Procmail 的规则要规定呢！关于原理部分我们就不再提了，有兴趣的回到上一篇去瞧一瞧，这里仅介绍作法喔！

关于 Open Relay Data Base

启动 ORDB 的功能只要：

1. 设定 /etc/postfix/main.cf
2. 重新加载 postfix

就可以啦！我们可以这样做：

```
[root@test root]# vi /etc/postfix/main.cf
smtpd_client_restrictions = hash:/etc/postfix/access,
     reject_rbl_client   relays.ordb.org,
     reject_rhsbl_client dsn.rfc-ignorant.org
# 还记得 main.cf 的语法喔！呵呵！上面说的是，我们可以利用/etc/postfix/access
# 以及 relays.ordb.org 以及 dsn.rfc-ignorant.org 等机制来抵挡黑名单主机之意
# 不过，如果您使用的是旧的 postfix 版本(1.5.xx)，那么可能您的设定需要改为
maps_rbl_domains = relays.ordb.org
smtpd_recipient_restrictions = permit_mynetworks, permit_sasl_authenticated,
     reject_unauth_destination, reject_maps_rbl

[root@test root]# /etc/rc.d/init.d/postfix reload
```

还有更多的 Open Relay 的机制可以参考这个网页喔：
http://jimsun.linxnet.com/misc/postfix-anti-UCE.txt

关于 Procmail 用法

在 Postfix 当中启用 procmail 真是太简单啦！同样的设定只要一行就够了！

```
[root@test root]# vi /etc/postfix/main.cf
mailbox_command = /usr/bin/procmail

[root@test root]# /etc/rc.d/init.d/postfix reload
```

就是这么简单！至于 procmailrc 这个档案，请参考 sendmail 的简易设定一文喔！不过，事实上，使用 Procmail 在 Postfix 上面属于额外新增一项功能的设定，其实我们可以使用 Postfix 预设的邮件过滤功能即可，那就是使用信件的 header 与 body 过滤机制来达成啰！至于作法我们在底下进行介绍啰！

关于邮件过滤的规则设定

相信您如果架设过邮件主机的话，一定有曾经 (1)被广告信件轰炸与 (2)被莫名的病毒信件攻击

的窘境发生过～真的是很讨厌！我们一般的作法有可能是利用额外的 Procmail 来进行邮件的分析与过滤，这是一个不错的选择啦！不过，由于使用 Procmail 时，他是另外一支程序，所以可能会造成程序与登录文件属性不合的问题发生呢！那怎么办呢？呵呵！其实在 Postfix 里面他原本就预设了两个邮件过滤的机制了，分别是标头(Header)与内容(Body)这两部份的过滤机制喔！

先来说明一下，什么是 Header 与 Body 呢？最简单的想法你可以这样想：在你接到的一封信中，这封信主要分为底下这两部份：

- 信封上面的信息 (Header)：这包含了寄件者、收件者、地址、与信件标题等等！至于在 E-mail 上面，就是所谓的标题 (Subject) 、送件者 (From:)、收件者 (To:) 以及其它相关的信息等等啰！

- 信封内的信纸信息 (Body)：这就是对方寄信给你的时候所书写的内容啦！

如果我们能够针对这两者来进行规则的过滤，如此一来，呵呵！当然就可以抵挡掉大部分的问题信件咯！这些问题信件主要就是病毒信与广告信了。不过，您必须要了解的是，这个过滤的规则是以正规表示法 (Regular Expression, RE) 来进行设定的！因此，您最好能够具有初阶的 RE 概念，如此一来会比较容易看懂底下的数据喔！

5. 启用 Postfix 设定档里头的设定：
   好了，首先我们当然必须要启动设定档里面针对这个邮件过滤的设定了！这里我们订定为这样：
   - 对于 Header 的过滤规则以 /etc/postfix/header_checks 来进行设定；
   - 对于 body 的过滤规则以 /etc/postfix/body_checks 来进行设定。

   然后下达这样的参数：

```
[root@test root]# vi /etc/postfix/main.cf
# 在这个档案的最底下加入这两行
header_checks = regexp:/etc/postfix/header_checks
body_checks = regexp:/etc/postfix/body_checks
# 注意一下，那个 regexp 表示『我使用正规表示法来进行过滤的规则订定』的意思
# 至于后面接的档案就是设定档啦！

[root@test root]# postfix reload
```

6. 开始设定规则：

对于规则的订定其实并不简单的！我们有必要针对 RE 的规则来进行约略的说明：

- 在规则设定文件里面 (就是 header_checks 与 body_checks) 只要是 # 代表该行为批注，系统或直接略过；
- 所谓的过滤规则即是 Header 与 body 里面的『关键词』，例如我不想让 192.168.100.5 这个 IP 寄件到我的 mail server ，那么这个 From:.* 192.168.100.5 就是一条规则了！那个 .* 代表什么意思呢？他代表『没有或多个任意字符』的意思～更详细的说明请参考各个正规表示法的标准说明了！常见的正规表示法特殊字符与意义为：

  『.』：代表任意字符
  『\』：代表跳脱字符，可以让后面接的一个字符变成一般字符；
  『*』：代表重复零个或多个前一个 RE 的字符，例如『.*』则代表任意零个或多个字符的意思；
  『^』：代表『这一行的第一个字符需要符合规则』的意思；
  『$』：代表这一行的最后一个字符必须要符合这个字符的意思，

- 单一规则的设定方法为：

  /规则/ 动作 显示在登录文件里面的讯息

  请注意，要使用两个『 / 』将规则包起来喔！举个例子来说明：例如我想要 (1) 抵挡掉标题为 A funny game 的信件，(2)并且在登录文件里面显示 drop header deny，我可以这样写：

  ```
  /^Subject:.*A funny game/  DISCARD drop header deny
  ```

- 在预设的规则当中，大小写是视为相同的；
- 如果有两条以上的规则，那么就必须要使用 if 了，例如底下的案例：

  ```
  if /^Content\-Type:.*audio.*x\-midi/
  /^.*name\=.*\.scr/  DISCARD   drop the header inavalid
  endif
  ```

  上面的意思是，当一封邮件里面同时包含『 Content-Type: audio.x-midi 』与『name=*.scr』时，该封信件就会被丢弃了！那么如果有三条以上的规则时呢？呵呵！就是使用多个 if 来进行啦！

  ```
  if /rule1/
  if /rule2/
  /rule3/  动作    显示字眼
  endif
  endif
  ```

不过请特别留意，这个 if .... endif 的设定我仅在 2.x 版本上面试过，是没有问题的，不过，已经有很多的朋友提出说，在 1.xx 版本上面执行时会有问题发生，所以如果您的 Postfix 不是 2.xx 版本，那么底下鸟哥列出的两个范例就参考看看即可，不可直接套用喔！

- 关于动作有底下几个动作：

  REJECT ：将该封信件退回给原发信者；
  WARN ：将信件收下来，但是将该封信的基本数据记录在登录文件内；
  DISCARD：将该封信件丢弃，并不给予原发信者回应！

一般来说我是比较喜欢以 DISCARD 将信件直接丢弃的啦！ ^_^

7. 此外，请特别留意，在各主要 Linux distribution 释出的 1.xx 版本中，并无法使用 DISCARD 的规则喔！所以您只能使用 REJECT 了！底下列出鸟哥的两个范本：
   header_checks 范本
   body_checks 范本

8. 检查规则档案设定是否正确：
   请注意！不是设定好了就 OK 了！你必须要检查一下刚刚设定的规则是否正确？请特别留意喔！如果设定错误的话，很有可能会造成邮件无法顺利的被你的 Mail server 接收下来的困境呢！检查的方法很简单的，利用 postmap 来检查即可！如下所示：

```
[root@test root]# postmap -q - regexp:/etc/postfix/header_checks < \
> /etc/postfix/header_checks
```

9. 上面这些字眼『postmap -q - regexp:/etc/postfix/header_checks < /etc/postfix/header_checks』是同一行喔！如果屏幕上没有出现任何的讯息，就表示至少您的规则订定没有疑问了!然后不需要重新启动 postfix ,刚刚的设定立刻生效啦!

藉由一个这样简单的邮件过滤机制，您就可以轻易的设定个人的邮件规则，并且将他抵挡在你的 mail server 之外呢！很不错吧！此外，你可以在接收完信件之后，如果有发现任何不满意的邮件时，想要将他过滤掉，那么：

10. 开启该不满意的邮件，并且进入查看『邮件原始档』的内容，找寻出该邮件的『关键词』；
11. 查寻一下该关键词是在 Header 还是在 Body 呢？
12. 将该规则加入 header_checks 或者是 body_checks ；
13. 以 postmap 检查一下该规则是否设定无误，如果显示出错误讯息请持续修改至无错误为只；

这样就 OK 啦！简单的很～ ^_^

问题信件的送达 notify_classes

如果你的 postfix 发生了邮件或者其它方面的问题，应该通知谁呢？预设情况下， Postfix 会通知 postmaster 这个人的，所以，你必须要在 aliases 这个档案里面设定 postmaster 对应的实体用户才行！一般来说，预设的 postfix 已经设定好 postmaster 的邮件会转交给 root 了！所以这里还可以比较不用理他！重要的是，我们必须要将哪些讯息送给 postmaster 呢？有底下这几样：

- bounce：将无法寄出的信件复制一份给 postmaster 啰！不过，为了寄件者的隐私，postmaster 接到的是已经去除原始标头(headers)的邮件；
- 2bounce：将两次无法寄出的邮件复制一份给 postmaster ；
- delay ：将延误寄出的信件的标头(headers)通知 postmaster ；
- policy：客户端的寄件需求被 postfix 订定的规则所拒绝时，发送错误的讯息给 postmaster 观察用！
- protocol：当由于 client 端或者是主机端因为执行某些程序，造成不完整的执行程序时(就是有错误发生啦！)，则通知 postmaster 一个协议错误的讯息(protocol errors)；
- resource：当无法寄出的邮件是由于本身 postfix 的资源(resource)所造成的，例如 queue 档案无法写入的错误讯息等，则通知 postmaster 该问题！
- software：由于相关软件的问题造成无法寄出信件时的通知！

一般来说，默认值是以 notify_classes = resource, software 来设定的！如果我们仅只要这样的功能，那么就直接将『notify_classes = resource, software 』加在 main.cf 当中吧！

其它应用说明：
在其它应用方面，基本上，与 sendmail 相似的，我们都需要：

1. 进行备份；
2. 进行磁盘配额限制(quota)；
3. 进行登录文件的查询与记录

这些动作与 sendmail 都很类似啦！所以您可以回上一章去参考一下喔！那么万一是无法使用 postfix 来寄信呢？你可以这样试看看：

1. 回归到最原始的状态，也就是不开放任何 SMTP, Open Relay, procmail 等等，以最原始的 main.cf 档案进行 postfix 的运作，然后检查一下重要的 myhostname, mydestination 以及 mynetwork 的设定，来看看 postfix 是否工作的很顺畅，然后再来一个一个的启用其它相关的控制喔！
2. 万一无法进行 SMTP 时，请确认您的 cyrus SASL 函式库是第几版的，然后在依照本文上面的设定来设定看看，应该不成问题吧！
3. 还是无法解决您的问题时，请检查一下您的 /var/log/maillog 这个登录档，问题的解决都在里面啊！！！

另外，我们的网友 yangsman 提供了一支不错的 script 用来启动 pwcheck 之用，有兴趣的朋友也可以参考喔！ script 内容如下：

```
[root@test root]# vi /etc/rc.d/init.d/pwcheck
#!/bin/sh
#
# Written By Yangsman 2003/05/5
#
#
# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} =  "no" ] && exit 0

[ -f /usr/sbin/pwcheck ] || exit 0

# See how we were called.
case  "$1" in
 start)
    # Start daemons.
    if [ ! -f /var/lock/subsys/pwcheck ]; then
    gprintf  "Starting pwcheck:  "
    /usr/sbin/pwcheck 2>/dev/null
    echo pwcheck
    touch /var/lock/subsys/pwcheck
    else
    $0 status
    fi
    ;;
 stop)
    # Stop daemons.
    if [ -f /var/lock/subsys/pwcheck ]; then
    gprintf  "Shutting down pwcheck:  "
    kill `ps -aux|grep -v ps |grep  "/usr/sbin/pwcheck"| \
        gawk '{print $2}'` 2>/dev/null
    echo pwcheck
    rm -f /var/lock/subsys/pwcheck
    else
    $0 status
    fi
```

```
    ;;
 restart)
    $0 stop
    $0 start
    ;;
 status)
       if [ -f /var/lock/subsys/pwcheck ]; then
         gprintf  "pwcheck (pid \c"
         gprintf  "`ps -aux|grep -v ps |grep  "/usr/sbin/pwcheck"| \
            gawk '{print $2}' 2>/dev/null`\c"
         gprintf  ")is runing ...\n"
    else
         gprintf  "pwcheck is't stopped \n"
    fi
    ;;
 *)
    gprintf  "Usage: %s {start|stop|restart|status}\n"  "$0"
    exit 1
esac

exit 0
```

参考资源

- Postfix 官方网站：http://www.postfix.org
- Cyrus-SASL 官方网站： http://asg.web.cmu.edu/cyrus/download/sasl/doc/
- Open Relay Database： http://www.ordb.org/

本章习题练习 (要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- 请问 Cyrus SASL 在 1.5.xx 以及 2.xx 版本中，用来作为 SMTP 的认证的机制有何不同？并请说明不同的版本与 Postfix 的搭配情况。
- 如果要让 Postfix 可以收发来自非本机的外部信件，您可以修改 main.cf 里面的什么参数？
- 如何察看您目前的 Postfix 服务器的所有设定参数？(使用什么指令？)
- 在 Postfix 当中，由于已经具有过滤邮件的机制，所以不太需要使用 procmail 了！请问，我该如何启用信件的 Header 过滤机制？同时，如何设定规则，使得 192.168.100.100 这个主机的来信，以及只要邮件的标头为『 Your account 』的信件就予以丢弃？

简易 FTP Server 架设 -- Wu FTP

最近更新日期：2003/09/13

在 Internet 上面的传输协议当中， FTP (File Transfer Protocol) 算是一个最古老的协定之一了！早期还没有这么多好用的协议之前(例如 SAMBA)，都是使用 FTP 来进行数据的传递的呢！另外，一般来说，数据的传输以 FTP 这个协议来传送是相当的快速的，而且某些场合当中其实也相当的方便。不过，值得注意的是，使用 FTP 来传输时，其实是具有一定程度的『危险性』，因为数据在 Internet 上面是完全没有受到保护的『明码』机制！所以，其实鸟哥不太建议大家使用这个服务器的啦！尤其当你建置好了 FTP 之后，如果经常进行数据的传递，对于您的网络频宽『真的有很大的损害』啊！不过，无论如何， FTP 仍然有其存在的必要！例如中山大学的 FTP 站就作的相当的棒啊！所以，底下我们就来谈一谈用在这个古老的协议上面，最古老的服务器之一： Wu FTP

参考资源
本章习题练习

原理：

File Transfer Protocol（FTP）是相当古老的网络协议之一，他最主要的功能就是进行 Server 端与 Client 端之间的档案传送的功能啦！FTP 其实是以 TCP 封包的模式进行 Server 与 Client 之间的联机，当联机建立之后，使用者可以在 Client 端连上 Server 端进行档案的下载与上传，此外，还可以直接管理用户在 Server 上面的档案呢，相当的方便！而这个最古老的 FTP 服务器软件，大概要算是 Wu FTP 了，所以，底下我们将针对 Wu FTP 进行设定的说明喔！

FTP 的功能简介

FTP 主机除了单纯的进行档案的传输与管理之外，其实他还提供了几个主要的功能，底下我们约略的来谈一谈：

1. 不同等级的使用者身份：FTP 预设的情况下可以提供三种主要的身份，分别是(1)实体账号,real user；(2)访客，guest；(3)匿名登入者，anonymous 这三种。分成三种身份主要可以做为主机的控管上面的便利性，而且也可以将使用者作一个有效的管理呢！例如实体用户可以进行的动作可能会比较多一些，至于匿名登入者，大概我们就仅提供他下载一下资源而已，并不许匿名者使用太多主机的资源啊！当然，这三种人物能够使用的『在线指令』自然也就不相同啰！ ^_^

2. 命令记录与登录文件记录：FTP 可以利用系统的 syslogd 这个 daemon 来进行数据的纪录，而记录的数据包括了使用者曾经下达过的命令与使用者传输数据(传输时间、档案大小等等)的纪录呢！

3. 限制或解除使用者家目录所在(change root, 简称 chroot)：为了避免使用者在您的 Linux 系统当中随意逛大街，意指离开使用者家目录而进入到 Linux 系统的其它目录去，所以将使用者的工作范围『局限』在使用者的家目录底下，嗯！实在是个不错的好主意！FTP 可以限制使用者仅能在自己的家目录当中活动喔！如此一来，由于使用者无法离开自己的家目录，而且登入 FTP 后，显示的『根目录』就是自己家目录的内容，这种环境称之为 change root ，简称 chroot ，改变根目录的意思啦！这有什么好处呢？当一个恶意的使用者以 FTP 登入您的系统当中，如果没有 chroot 的环境下，他可以到 /etc, /usr/local, /home 等其它重要目录底下去察看档案数据，尤其是很重要的 /etc/ 底下的设定档，如 /etc/passwd 等等。这样他就有办法取得系统的某些重要信息，用来『入侵』您的系统呢！所以在 chroot 的环境下，当然就比较安全一些咯！

FTP 的运作流程与使用到的 port

FTP 正常情况下的联机方向：
我们在网络基础当中得知，TCP 这种封包由于需要经过 Server 端与 Client 端两边的『三向交

握』之后，才能确定联机，因此，他可以说是一个比较『可靠』的联机模式，因为双边都已经经过确认(ACK)的动作，所以，当然会较为『可靠了』！那么既然 FTP 主要的工作是让 Client 与 Server 端可以进行档案的传输，自然需要较为可靠的联机啰，不然档案数据传到一半竟然损毁时，怎么办？！因此，FTP 当然就是以 TCP 这种封包来进行联机的。在这里，我们不厌其烦的，再次说明一下 FTP 在正常模式情况(或者称为主动模式，active)下，主机与 client 端到底是如何建立联机的呢(在不考虑防火墙与其它不知名因素的情况下)？

4. Client 端主动向 Server 端发送联机需求：由于是客户端想要连上 FTP 主机，所以呢，当然联机的方向首先会由 Client 发起！此时，Client 端随机选取一个大于 1024 以上的 port 来主动的联机到 FTP 主机提供的 FTP 端口口(通常为 port 21)，而由于是主动的联机封包，所以这个联机会带有 SYN 的标志在；

5. Server 端接受后，响应给 Client 端：当 Server 接收到 Client 的要求之后，会响应 Client 端的需求，此时 Server 端会建立等待联机的资源，并且将一带有 SYN 与确认 (ACK) 的封包送回 Client 端；

6. Client 端回应确认封包：在 Client 端接收到来自 Server 端告知的封包后，会再次的发送一个确认封包给主机，此时，两边才会正式的建立起联机的通道，这个步骤 1~3 就是 Three-Way Handshake(三向交握的啦!)。需要注意的是，这个已经建立联机的通道(通常是 port 21)仅能进行 FTP 的『指令』而已，如果该指令涉及到数据的传送(data transfer)时，例如上传或下载等等，那么就需要额外建立一条数据传输的信道才行 (ftp-data)！而数据传输的信道建立则需要继续底下的步骤；

7. Client 端发送数据传输要求的命令给 Server：当需要进行数据的传输时，Client 端会启用另一个高于 1024 的埠口来做为联机的准备(这个高于 1024 的埠口与步骤 1 那个埠口不是同一个！)，并且 Client 端会主动的利用刚刚已经建立的指令信道(通常是 port 21)发送一个命令告诉 Server 说：『我已经准备好一个数据传输的端口口了，请准备进行传输吧』！特别留意喔，这个时候 Client 是透过『命令通道』来对 Server 下达命令的，而且已经通知 Server 我(client)要启用的埠口了喔！

8. Server 端以 ftp-data 埠口主动联机到 Client ：收到命令之后的 Server 会『主动』的以 ftp-data 埠口(一般为 port 20)向 Client 端通知的那个高于 1024 的埠口进行联机，特别需要留意的是，此时是『Server 端主动向 Client 端的联机』喔，所以该联机的 TCP 封包会带有一个 SYN 的标志在；

9. Client 端响应主机端，并继续完成三向交握：在接到 Server 来的封包之后，Client 会响应一个带有 ACK 确认的封包，并继续来完成另一个三向交握的程序，此时，数据传输的信道才正式的建立。

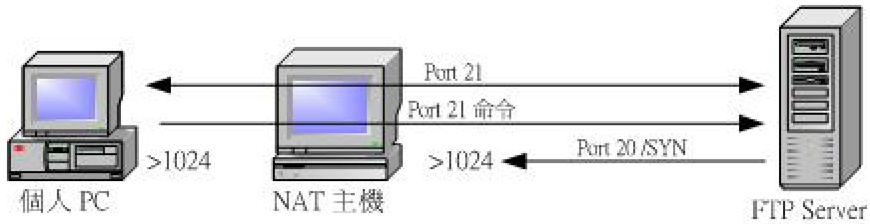
如此一来则成功的建立起『命令』与『数据传输』两个信道！不过，要注意的是，『数据传输信道』是在有数据传输的行为时才会建立的通道喔！并不是一开始连接到 Server 就立刻建立的通道呢！留意一下啰！

使用到的 port：
由上面的联机数据来看，其实我们会用到的主机的两个端口口分别是：

(1)命令通道的 ftp (预设为 port 21) 与
(2)数据传输的 ftp-data (预设为 port 20)。

这两个埠口的工作是不一样的，首先，那个 port 21 主要是用在接收 Client 端下达的命令之用，例如显示目录内的档案内容 dir 以及上传下载 (put, get) 等等的指令的；至于 port 20 刚刚上头约略提过了，就是用在数据传输的时候才会建立的一个联机呢！而且，重要的是，两者的联机方向是不一样的！首先， port 21 主要接受来自 Client 端的主动联机，至于 port 20 则为主动联机至 Client 端呢!这样的情况在 Server 与 Client 两者同时为公共 IP (Public IP) 的 Internet 上面，通常没有太大的问题，不过，万一你的 Client 端是在防火墙后端，或者是 NAT 主机后端呢？会有什么问题发生呢？底下我们来谈一谈这个严重的问题！

在 NAT 或者防火墙后端的 FTP Client 联机问题：
万一你的 FTP client 是在 NAT 主机的后端，那由于我们的 NAT 主机会自动的纪录 client 端向外联机的信息，所以在 Client 依上面步骤 1 送出要求封包后，步骤 2 的 FTP 回传的封包可以透过 NAT 转交给 client ，这没有问题!所以，Client 连接到 Server 的命令通道 (port 21) 可以正确的被建立起来的。不过，万一 Client 端在建立起了命令通道之后，对 Server 下达数据传输的命令呢？联机会是如何？我们以底下的图示来说明好了：



10. 由于目前的 NAT 主机可以记录由内部计算机联机出去的信息，因此，藉由 port 21 的联机可以顺利的被建立起来；
11. 当 Client 端由 port 21 下达数据传输的命令时，此时『个人 PC』会告诉 FTP Server 说：『我开了一个 >1024 的埠口等你来联机喔』！
12. 这个时候要特别留意的是，『个人PC』经过NAT主机联机后，在 FTP Server 看到的『个人PC』的 IP 其实是 NAT 那部主机的!所以，这个时候 FTP Server 会主动的由 port 20 向 NAT 主机的 >1024 那个 port 要求建立联机! (请翻阅NAT主机一章)

了解问题的所在了吗？原本我们的 FTP 主机要联机的其实是『个人 PC』这部计算机的 >1024 那个 port ，不过，由于 NAT 的关系，所以却造成了联机是向 NAT 主机的 >1024 那个 port 进行主动联机的！如此一来，想当然尔，NAT 主机并没有启动 >1024 那个 port 等 FTP 主机来联机，所以自然无法成功的建立起联机，这个时候你就会看到『Can't build data connection: Connection refused, 无法进行数据传输』之类的讯息了！啊！真是惨啊！ @_@

那有没有办法可以克服这个问题呢？难道真的在 NAT 主机后面就一定无法使用 FTP 吗？当然不是！目前有两个简易的方法可以克服这个问题：

13. 使用 Linux NAT 主机的 iptables 预设模块，亦即 ip_conntrack_ftp 与 ip_nat_ftp 这两个核心模块！应用 modprobe 这个指令就可以将这两个模块加载了！我们刚刚说过，既然 iptables (NAT 主机)可以记录 client 端向外联机的信息，而 client 端向 server 端要求数据传输时，会主动告知 Server 我(client) 要等你来联机的 port ，因此，这两个聪明的模块可以透过检查 NAT 主机内的信息而加以应用，那就可以让 Server 与 Client 端建立 ftp-data 的联机啦！不过，这两个模块并不是万能的，因为这两个模块目前仅能针对预设的 ftp-data(port 20) 进行检验的工作，万一您联机的是一个使用非 port 20 为 ftp-data 传输的主机时，那这两个模块就没有办法发挥其效能了！

14. 另一个方法就是使用被动式联机 (passive)！什么是被动式联机呢？想一想，既然『主机主动』联机到我的 NAT 后面的 client 不能成功，那么我反其道而行，如果以 client 来连到 server 呢？是否就变成如同 port 21 相似的联机方向，如此一来不就可以成功的建立联机了吗？呵呵！没错，而且也不需要启动上面两个模块了，并且也不担心 FTP Server 是否启用非 port 20 的 ftp-data port 啦！这部份我们底下说明喔。

什么是『被动, passive』的联机

既然在 NAT (或防火墙) 后端的 client 无法让主机主动的来建立联机，那么我就让主机『被动的等我 client 去联机』啊！果真如此的话，那么我的联机状态变成如何了呢？

15. Client 端主动向 Server 端发送联机需求：

16. Server 端接受后，响应给 Client 端：

17. Client 端回应确认封包：上面这三个步骤与主动式联机一样，同样的完成三向交握后，建立命令通道了！底下说明被动式数据传输信道的建立。

18. Client 端发送数据传输要求的命令给 Server：与主动式联机不一样的是，在被动式联机时，Client 端在下达命令之后，并告诉 FTP Server 说『我要使用 PASV 模式(就是 passive 啦！)的方式来进行数据传输』；

19. Server 端挑选 ＞1024 的埠口等待联机：在接受 client 的 PASV 要求之后，如果没有特别的设定时 (目前的 FTP 服务器版本已经可以指定 passive port 来规定被动式连接的端口口号码) ，Server 会随机选取一个大于 1024 的埠口，并经由命令通道告诉 client 端说：『我已经开了一个 ftp-data 的埠口等妳联机喔!』并开始等待 client 端的联机；

20. Client 端主动向 Server 端建立联机并继续完成三向交握：经由命令通道得知 Server 的端口口之后， Client 端会随机挑选另一个大于 1024 的埠口，并主动向 Server 端的等待联机的埠口进行联机动作，所以此联机封包是带有 SYN 的标志的喔!然后 Server

会响应一个带有 ACK 确认的封包，并继续来完成另一个三向交握的程序，此时，数据传输的信道就正式的建立。

发现上面的不同点了吗？呵呵！如此一来，在 NAT 主机内部的 Client 就可以顺利的连接上 FTP Server 了！这就是在 FTP 联机里面的所谓被动式联机啰！但是，万一 FTP 主机是在 NAT 后端那怎么办.....呵呵！那可就糗了吧～ @_@这里就牵涉到更深入的 DMZ 技巧了，我们这里暂不介绍这些深入的技巧，先理解一下这些特殊的联机方向，这将有助于您未来服务器架设时候的考虑因素喔！

此外，不晓得您有无发现，呵呵！透过 PASV 模式，Server 在没有特别设定的情况下，会随机选取大于 1024 的 port 来提供 Client 端连接之用。那么万一主机启用的 port 被搞鬼怎么办？而且，如此一来也很难追踪来自入侵者攻击的登录信息啊！所以，这个时候我们可以透过 passive ports 的功能来『限定』主机取用的 port number 喔！

FTP 的安全性问题与替代方案

事实上，FTP 是一个不太安全的协定呢！怎么说呢？很简单啊！因为 FTP 与 Telnet 相似的，他是以『明码』的状态在 Internet 上面流窜的，所以当然就容易被有心人士将你的数据给他抓下来，并且加以利用啦！因此，他当然不是很安全啊！所以，在网络上大家才会常常告诫说，不要随意架设 FTP 网站啊！否则主机怎么被破解的都不晓得哩！此外，由于 FTP 软件常常会有漏洞的问题，因此也要常常更新套件喔！另外，其实拜 SSH 所赐，目前我们已经有较为安全的 FTP 了，那就是 ssh 提供的 sftp 这个 server 啊！这个 sftp-server 最大的优点就是：『他是经过加密的资料！』所以在 Internet 上面流窜的时候，嘿嘿！毕竟是比较安全一些啦！所以，建议您，除非必要，否则的话，使用 SSH 提供的 sftp-server 功能即可～然而这个功能对于一些习惯了图形接口，或者是有中文档名的使用者来说，实在是不怎么方便，因为目前还没有很棒的 sftp-server 的图形接口软件说～所以，有的时候，FTP 站还是有其存在的需要的。如果真的要架设 FTP 网站，那么还是得需要注意几个事项喔：

21. 随时更新到最新版本的 FTP 软件，并随时注意漏洞讯息；
22. 善用 iptables 来规定可以使用 FTP 的网域；
23. 善用 TCP_Wrappers 来规范可以登入的网域；
24. 善用 FTP 软件的设定来限制使用您 FTP 主机的使用者的不同权限啊；
25. 使用 Super daemon 来进阶管理您的 FTP 主机；
26. 随时注意使用者的家目录、以及匿名使用者登入的目录的『档案权限』；
27. 若不对外公开的话，或许也可以修改 FTP 的 port 。

无论如何，在网络上听过太多人都是由于开放 FTP 这个服务器而导致整个主机被入侵的事件，所以，这里真的要给他一直不断的强调，要注意安全啊！

什么时候才要设定 FTP 啊！开放谁人连进来

既然(1)FTP 不怎么安全(2)FTP 的使用者身份至少有三种，那么在设定 FTP 服务器之前就需要针对这些不同身份者的登入限制来做规划啰！以达成较为安全的管理啊！底下我们谈一些大致的概念性问题，这些真的是蛮概念的～看看即可！

开放的用户身份与可能造成的危害	建议事项
实体用户(Real user) o 在预设的条件下，开放 FTP 本来就提供了实体用户登入之用。 o 不过，需要了解的是，以实体用户做为 FTP 登入者身份时，基本上，系统并没有针对实体用户来进行『限制』的，所以他可以针对整个档案系统进行任何的工作。因此，如果您的 FTP 使用者没能好好的保护自己的密码，导致被入侵，那么你的整个 Linux 系统将很有可能被毁灭啊！	o 由于实体用户本来就可以透过网络连接到主机来进行工作，因此实在没有特别的需要开放 FTP 的服务啊！例如 sftp 本来就能达到传输档案的功能啰！ o 如果确定要让实体用户使用者利用 FTP 服务器的话，那么您就需要避免让几个系统用的账号可以登入!这个时候可以将『不想让他登入』的账号写入 /etc/ftpusrs 这个档案当中啊!例如 root 就是个很好的例子！
访客(Guest) o 通常会建立 guest 身份的案例当中，多半是由于主机提供了类似『个人 Web 首页』的功能给一般身份使用者，那么这些使用者总是需要管理自己的网页空间吧？这个时候将使用者的身份压缩成为 guest ，并且将他的可用目录设定好，即可提供使用者一个方便的使用环境了！且不需要提供他 real user 的权限喔！	o 仅提供需要登入的账号即可，不需要提供系统上面所有人均可登入的环境啊！ o 当然，我们在主机的设定当中，需要针对不同的访客给他们不一样的『家目录』，而这个家目录与使用者的权限设定需要相符合喔!例如要提供 test 这个人管理他的网页空间，而他的网页空间放置在 /home/test/www 底下，那我就将 test 在 FTP 提供的目录仅有 /home/test/www 而已，比较安全啦！而且也方便使用者啊！ o 针对这样的身份者，需要设定较多的限制，包括：上下传档案数目与硬盘容量的限制、联机登入的时间限制、许可使用的指令要减少很多很多，例如 chmod 就不要允许他使用等等！
匿名者(anonymoust) o 提供匿名登入实在不是个好主意～因为毕竟你的系统为何要让别人登入利用呢？ o 不过，如果是提供整个学校单位来利用的话，那就另当别论了！	o 无论如何，提供匿名登入都是一件相当危险的事情，因为，只要您一不小心，将重要的资料放置到匿名者可以读取的目录中时，那么就很有可能会泄密！与其战战兢兢，不如就不要设定啊～ o 果真要开放匿名登入时，很多限制都要进行的，这包括：(1)允许的工作指令要减低很多，几乎就不许匿名者使用指令

	啦、(2)限制档案传输的数量，尽量不要允许『上传』数据的设定、(3)限制匿名者同时登入的最大联机数量，可以控制盗连喔！

套件安装：

事实上，使用 Wu ftp 来架设你的 FTP 服务器时，还是以 RPM 的方式来安装比较好啦！另外，如果您的 Linux distribution 提供其它版本的 FTP 服务器，呵呵！那么就不要使用 wu ftp 也没有关系啊！这是因为 wu ftp 实在是太古老了，所以很多的黑客软件都是针对他来设计的，也因为如此啊，所以才会产生『Wu FTP 服务器比较不安全』的情况啊！好了，底下我们以 Red Hat 7.x 的版本来进行说明吧。基本上，一个 FTP 服务器包含 Server 与 Client 用途的套件至少要有：

```
[root@test root]# rpm -qa | grep ftp
ncftp-3.0.3-6
ftp-0.17-12
wu-ftpd-2.6.1-20
```

其中：

- wu-ftpd ：这就是主要的 FTP 服务器套件啦！
- ftp ：提供 ftp 指令，就是 client 端的工具啦！
- ncftp ：提供匿名登入的 FTP 网站的 client 端相当棒的另一套联机 FTP 软件！

如果没有安装，请马上安装吧！如果不晓得怎么安装，那么请拿出『鸟哥的 Linux 私房菜 -- 基础学习篇』好好的将 mount CD 的指令、搜寻的指令，以及 RPM 的指令瞧一瞧先！

Server 端设定：

其实 Server 端的设定最主要的就仅有 /etc/ftpaccess 这个主要设定档啦!几乎只要他搞定了，FTP 就不会有问题说～不过，由于 FTP 的高危险性，所以其它几个跟安全较有相关的档案我们也得来瞧一瞧才行啊！因此上，我们首先就需要来了解一下 wu ftp 到底有哪些设定档案与执行档呢？

Wu FTP 的结构

Wu FTP 的档案结构先来了解一下，才好继续进行说明啊！底下主要粗分为设定档与执行档进行说明喔！

设定档： Wu FTP 的设定档主要有底下这几个：

- /etc/ftpaccess：这是最主要的设定档了！所有跟 Wu FTP 有关的设定内容，都可以在这个档案做修订；

- /etc/pam.d/ftp, 与 /etc/ftpusers：这两个档案与 PAM 模块关系较大！在预设的情况中，只要在 /etc/ftpusers 这个档案内的使用者『都不能使用 FTP 的服务』
- /etc/ftphosts：用来允许或拒绝(allow/deny)某部主机或者某位使用者是否能够登入 FTP 主机的设定档案，基本上，这档案里面的设定也可以直接在 /etc/ftpaccess 当中设定喔！
- /etc/xinetd.d/wu-ftpd：这个是用来启动 FTP 的 daemon 设定档案～当然啦，主要是挂在 xinetd 这个 daemon 下的，如果是挂在 inetd 这个 daemon 时，就有可能是 /etc/inet.d 底下的档案啰！

执行档：除了上面提到的这些设定档之外，还有一些执行档也需要了解一下：

- ftpcount ：主要用来计算『目前联机的人数』，可以计算出各种身份的联机人数啊！
- ftpwho：可以显示出『目前联机的使用者是那个 User ？使用那个 PID？动作多久了？』等等的信息呢！
- ftprestart：重新启动 ftp 啊！
- ftpshut：指定时候关闭 FTP 的一个指令喔！
- in.ftpd：这个就是主要的 Wu FTP 的 daemon 啰！我们启动的 wu ftp 就是他的工作呢！

客户端的使用执行档：这个部分的指令并不是 wu ftp 所提供的，但是粉重要，所以先提出说明喔！

- ftp：就是最阳春的 client 端软件啰！
- ncftp：可以使用在匿名 FTP 网站喔！相当棒的软件！可以支持整个目录的下载呢！

呵呵！接下来就是那个主要的 FTP 设定档的设定部分啦！

最简单的 ftpaccess 设定档

事实上，与 Wu FTP 关系最大的就是 /etc/ftpaccess 这个档案啦！只要他设定好，其它的地方相对都不成问题的！而其实 Wu FTP 一开始已经帮我们设定好一个最简单的 ftpaccess 档案，我们先来谈一谈这个档案的几个主要的设定项目，然后再来继续其它的设定项目呢！

```
[root@test root]# vi /etc/ftpaccess
# 1. 设定人物群组名称
#    设定这个 FTP 服务器的人物身份设定，使用 class 来设定的！他的语法是：
#    class  <人物群组名称>  <用户身份 1,用户身份 2,..>  <允许联机的来源>

class   all   real,guest,anonymous  *

# 上面的意思是说，我设定一个类别群组为 all ，这个 all  里面就包含了
```

```
# 三种身份的使用者，就是 FTP 预设的 real, guest 与 anonymous 这三个，
# 需要注意的是，这三个类别的使用者之间是以逗号『,』隔开的，并没有空格符
# 而这个 class 允许的来源来自任何地方『*』。
# 这个 class 可以多重设定，并且，万一重复设定时，以第一个出现的 class 类别
# 为准！举个例子，假如我的 FTP 里面的 real 仅允许学术网络登入，至于其它
# 的 guest 与 anonymous 则虽然可以由任何地方登入，但是不可以由 chinait.com
# 这个网域以及 61.141.0.0/16 这个网域登入时，那我可以这样设定两个 class 喔：
# class  allone  real,guest,anonymous  *.edu.tw
# class  alltwo  guest,anonymous    !*.chinait.com  !61.141.0.0/16  *
# 请注意到，惊叹号『!』有代表『否，不允许』的意思存在，而星号『*』则代表
# 任何地方的意思，则如上面所设定时，如此一来，学术单位可以连到我的 FTP ，
# 至于 guest 与 anonymous 则可以任何地方连进，当然，除了上面的两个网域之外
# 所以说，经由这个 class 的设定，就可以轻易的将三种身份是否可以登入主机的
# 状态搞定了！ ^_^

# 2. 设定系统的 FTP 管理员的 e-mail 信箱地址，与主机名称！
#    单纯的就是显示出系统当中 FTP 服务器管理员的网址啦！预设的设定如下：

email root@localhost
hostname vbird.adsldns.org

# 一般来说，我会将这个 e-mail 后面的地址写上可以被使用者发信的信箱，例如：
# email testing@test.adsldns.org
# 这样的格式！这个 email 可能会出现在进出网站时的欢迎画面当中！
# 最大的任务是：当使用者发现问题的时候，可以跟系统的管理员联络啊！
# 所以当然要写下『可以收信』的正常 email 啰！
# 至于那个 hostname 则仅与欢迎画面时的变数有关！

# 3. 允许同一次联机当中，错误登入的次数
#    为了避免被不明攻击者的『暴力攻击』法，所以在一次联机当中，
#    仅允许对方最多有 5 次的登入机会，如果密码或 ID 一直发生错误，
#    则会将该联机『踢』掉的啦！

loginfails 5

# 当然啰！如果您想将登入的次数改小一点的话，也可以使用『loginfails 3』

# 4. 向使用者显示『README, 读我』档案的内容讯息！
#    当使用者登入或者变换目录时，若目的端目录有 README 这个档案时
#    (可以附加文件名)，则向使用者显示该档案的内容！语法为：
#    <readme>  <README*>  <动作>
#    一般来说，动作有『登入』与『变换目录』，代号为 login 与 cwd=*

readme  README*    login
```

```
readme   README*     cwd=*

# 举个例子来说，我是 testing 这个身份的使用者，在我的家目录内有个档案：
# /home/testing/data/README.important
# 那么当我使用 FTP 软件连进我的家目录 (/home/testing) 然后切换目录到
# /home/testing/data 后，我的屏幕就会出现『请读取 README.important』
# 的字样啰！以提醒使用者之用！

# 5. 与 readme 的意义蛮相同的！不过这个 message 却会将后面所接的档案的
#     内容直接显示在屏幕上面，而不仅是告知使用者去读取而已～

message /welcome.msg            login
message .message                cwd=*

# 上面的意思是说，当我 login 或者切换到任何有档名为 .message 的目录时，
# 该档案的内容就会显示到屏幕上面！一般来说，那个 /welcome.msg 就是
# 『进站欢迎画面』啰！这个等一下我们在底下会独立出一小节来介绍他！

# 6. 是否提供使用者在线立即执行的指令！
#     一般的格式为：
#     <指令名称>  <是否允许/yes/no>  <针对的对象是谁>

compress        yes             all
tar             yes             all
chmod           no              guest,anonymous
delete          no              anonymous
overwrite       no              anonymous
rename          no              anonymous
umask           no              all

# 以上面的例子来说，我允许任何成功登入我主机的使用者(all)使用我的
# FTP 主机来执行压缩这个指令的动作！但是我不许匿名者(anonymous)
# 使用我的 FTP 主机进行删除(delete)以及改名(rename)的动作！
# 你当然还可以增加自己所想要提供，或者减少提供使用者使用的指令！
# 当然啦，既然 FTP 主要是针对『档案』，所以指令以档案的删除、移动、
# 更改与压缩为主！

# 7. 将使用者执行的部分指令历程记录到 /var/log/xferlog 这个档案
#     FTP 进行上传、下载或者其它使用者动作时，可以将过程讯息记录下来，
#     记录的档案就是 /var/log/xferlog 这个档案啰！语法为：
#     <log>  <欲登录的项目>  <记录的使用者身份>  <何种动作>

log transfers anonymous,guest,real inbound,outbound
```

```
# 上面说明的是『针对档案传输(transfers)进行记录，而针对所有人均纪录，
# 分别记录上传与下载(inbound,outbound)』，请注意，身份如果有多种，要以
# 逗号『,』隔开，不要加空白喔！所以，当你的 FTP 使用者连上主机，
# 并且有任何档案传输的动作时，则档案大小以及档案数等信息，就会被纪录
# 到 /var/log/xferlog 里面去啦！而除了档案传输之外，还有什么可以纪录的呢？
# 基本上，那个『欲登录的项目』内容就包含了下面几项资料：
# a. log commands <身份> ：例如『log commands real,anonymous』，表示
#     real 与 anonymous 这两种身份的人，在 FTP 上面所下达的任何指令君会
#     被纪录在 /var/log/xferlog 里面
# b. log security <typelist> ：例如『log security guest,anonymous』
#     表示当 guest 与 anonymous 使用者『违反安全机制』时，则会将当时
#     使用者所下达的指令或者其它动作纪录下来！

# 8. 关闭 FTP 的设定档！
#     我们可以设定关闭 FTP 这个服务的时间，就利用 shutdown 后面接的档案！

shutdown /etc/shutmsg

# 如果 /etc/shutmsg 不存在，则 FTP 服务就不会被关闭！所以不存在没关系！
# 而如果 /etc/shutmsg 存在的话，他的内容包含有底下这些资料(注意：
# 第一行为时间参数，共有七个时间参数，用空格键分隔，而提示文字可以随便
# 编写内容喔！也可以使用变量啊！)：
# <年> <月> <日> <时> <分> <抵挡新联机> <删除已联机>
# <提示文字>
# 年：任何大于 1970 年的年份；月：0-11！请注意啊！是由 0-11 喔！
# 0 代表 1 月、 1 代表 2 月！
# 日：当然就是 1-31 啰！      ；时：由 0-23 ；分：0-59
# 抵挡新联机与删除已联机：格式是 HHMM 例如 90 分钟则是 0130 ，在关机前的
# 设定时内，会拒绝新联机与将以联机之通道切除喔！例如：
# 『2003 5 30 12 0 0230 0030
#  I will shutdown my FTP server !sorry!』
# 在 2003/6/30 的 12:00 要关闭 FTP ，而 12:00 之前的两小时 30 分内(09:30)
# 就不许新的尝试登入的联机，而在 30 分钟前(11:30)就切掉已经已经联机之
# 使用者联机！事实上，这个 shutdown 蛮有趣的！因为实际上，您的 FTP
# 服务并没有关掉，仅只是让他人无法使用 FTP 而已啊！那么如何重新启动呢？
# 很简单啊！将 /etc/shutmsg 杀掉，或者里面的时间更动一下即可！

# 9. 匿名者的密码验证：
#     如果您的 FTP 允许 anonymous 的话，那么还是需要让匿名者输入密码的，
#     不过就是密码的设定比较松散就是了！目前的密码格式为：
#     <passwd-check>  <no|trivial|rfc822>  <动作>

passwd-check rfc822 warn
```

```
# 上面说的是，以匿名者登入的使用者也需要输入密码，而密码的格式为 rfc822，
# 如果使用者的密码不合格，那么就警告(warn)使用者，但仍允许使用者登入！
# 密码的格式方面目前有两种(no 是不需要密码确认，所以不讨论！)
# trivial：密码当中必须含有 @ 这个 e-mail 的字符；
# rfc822 ：密码必须符合 frc822 的规范！
# 通常我们使用的是 rfc822 即可！至于动作主要有两种动作：
# warn   ：使用者输入错误的密码时，仅显示警告讯息，仍允许其登入；
# enforce：使用者若输入错误密码，储显示警告讯息，并中断联机喔！
# 注意：
# 如果你不想让某个 email 的型态通过认证时，可以使用 deny-email 这个
# 项目来抵挡！举个例子来说，你不想让 IE 的预设邮件地址通过认证，可使用
# deny-email IE?0User@
# deny-email mozilla@
# 上面这两个项目可以同时存在，如果还有不想让他通过的 email address
# 可以持续上面的设定多行！这有什么用途呢？如果您不想让 web browsers
# 通过密码的确认，而仅想让类似一般的 FTP client 来联机，那么这个
# 限制项目就有用的很了！因为他可以将 IE 之类的 browsers 挡下来啊！

# 10. 设定允许与不许登入 FTP 服务器的使用者与群组

deny-uid %-99 %65534-
deny-gid %-99 %65534-
allow-uid ftp
allow-gid ftp

# 这个是在 Red Hat 系统上面新增出来的设定啦！在一般正常的系统当中，
# UID 小于 100 通常是系统账号，而 UID 大于 65534 可能有安全上的问题，
# 所以，我们就直接将这两段 UID 与 GID 切掉啊！让他们无法登入，也就可以
# 拒绝某些不当的入侵攻击了！那就是 deny-uid 与 deny-gid 的功效！
# 所以，上面的意义是，小于 99 与大于 65534 的 UID/GID 都予以抵挡联机；
# 而开放的 UID 与 GID 则仅有 ftp 这个群组与使用者喔！
# deny-uid 后面除了接数字外，也可以直接接账号名称，例如要挡住 testing
# 与 testqq 这个用户时，可以设定：
# deny-uid testing testqq
# 后面可以接多个 UID 或账号或者使用范围，例如抵挡 100 到 1000 之间的 UID
# deny-uid 100-1000
# 至于 allow-uid 则恰好相反啊！就是允许的意思～
```

事实上，上面的设定并不麻烦，只有几行而已，只是我们需要了解一下每个项目之间的相关性，所以鸟哥我加注了一些说明而已～您可以参考参考的啦！这样就完成了一个最最简单的 ftpaccess 这个设定档啰！准备来去启动 FTP 啰！

使用 Super daemon 管理 FTP 的情况

目前一般常见的 FTP 服务器(除非是大型的 FTP 主机)大多是使用 super daemon 来进行统一管理的，而由于目前多半的 super daemon 都使用 xinetd 这个 super daemon ，所以底下我们仅针对这个 xinetd 来进行说明喔！事实上，由于 FTP 是挂在 super daemon 底下的一个服务，所以我们仅需要设定好 xinetd 里面关于 wu-ftpd 的设定档，然后『重新启动 xinetd 』就可以启动 FTP 啰！在一般的情况下，我们可以发现 /etc/xinetd.d/wu-ftpd 这个主要设定档内容为：

```
[root@test root]# vi /etc/xinetd.d/wu-ftpd
service ftp
{
        disable = yes  <==就是他，将他改为 no 即可！
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/in.ftpd
        server_args             = -l -a
        log_on_success          += DURATION USERID
        log_on_failure          += USERID
        nice                    = 10
}
```

在上面粗体的地方将 yes 改为 no 即可！因为 disable 是『取消』的意思，那 disable = no 当然就表示『不取消』的意思～这是一个相当简单的 xinetd 的设定档，如果要进行多重控制的话，例如：我的主机有两块适配卡，一块对内一块对外，对内与对外的『开放时间』与『开放网域』及『相关权限』都不相同时，就可以使用其它额外的设定来控制这些参数！由于相关的说明我们已经在『鸟哥的 Linux 私房菜 -- 基础学习篇』里面的『认识服务』谈过多次，而且，在架站篇里面的 Telnet 服务器当中也提过一个简单的范例，所以这里就不再多作赘述，请自行前往查阅！

好了，那么最终终于要来启动 FTP 啰！赶紧动手重新启动 xinetd ，并且使用 netstat 来察看一下 port 喔！

```
[root@test root]# /etc/rc.d/init.d/xinetd restart

[root@test root]# netstat -tl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:ftp                   *:*                     LISTEN

[root@test root]# ftp localhost
Connected to localhost (127.0.0.1).
220 localhost.localdomain FTP server (Version wu-2.6.1-20) ready.
Name (localhost:testing): testing  <==输入登入者账号
```

```
331 Password required for testing.
Password:  <==输入你的密码
230 User testing logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> exit<==离开 FTP
```

这样就 OK 的啦！(注：还是得注意一下你的防火墙机制与 /etc/hosts.deny 里面是否将 in.ftpd 这个服务关掉了？！会造成无法联机的原因有很多都是因为没有将防火墙的机制打开的缘故！请再回头详细看一下简易防火墙设定一文)。如果要关闭 FTP 的话，那就将上面 disable = no 改成 disable = yes 然后再重新启动 xinetd 即可！相当的简单吧！ ^_^

欢迎画面的建立、 Readme 档案与关闭 FTP 讯息内容

欢迎画面的建立：

呵呵！建立进站欢迎画面可就快乐的很了！可以让大家知道您这位 FTP 主机管理员所下达的『公告』事项喔！我们刚刚在前头已经设定了进站欢迎画面(login)的档案是 /welcome.msg ，注意喔！是放在『根目录，root』底下，那么到底 / 是怎么定义的？是与 Linux 的档案系统相同还是有其它的规定！？是这样的：

- 如果您的使用者可以到达系统的根目录，那么他将取用 /welcome.msg 这个档案；
- 如果使用者只能在自己的家目录内活动时，那么此时他的『根目录』会变成他的『家目录』！因为这样他才离不开自己目前所在的目录啊！例如 test 这个使用者被限制在自己的家目录内活动，那么他的进站欢迎画面则是取用 /home/test/welcome.msg 喔！
- 至于如果是匿名者登入的话，由于匿名者一定都会被限制在匿名者登入的目录，所以您也必需要将该档案放置在他们的根目录喔！

那么由于 welcome.msg 里面其实包含有相当多的可用变量，包括目前的主机、远程主机名称、FTP 管理员的 email 以及目前时间等等的变量，我们先来谈一谈有哪些变量吧：

```
%T    本地端的主机时间(格式为 Fri Mar 21 11:28:50 2003)
%F    使用者目前所在目录之 partition 所剩空间(不一定支持所有系统)
%C    使用者目前所在的目录
%E    系统管理员的 email ，这个就是刚刚设定 ftpaccess 内的 email 值
%R    远程主机的 IP 或 hostname ！
%L    本地端主机的名称或 IP
%U    使用者的登入账号名称
%M    FTP 主机所能允许的使用者最大联机数量
%N    FTP 主机目前已经联机的使用者数量
%B    关于硬盘容量的限额
```

%Q	目前的 block 数量
%I	最大的可用 inodes
%i	针对 inodes 的限额
%H	当过度使用硬盘空间时的时间限制
%h	当使用过度档案时的时间限制
%s	预计关闭 FTP 的时间(与 /etc/shutmsg 有关)
%r	预计关闭 FTP 前禁止再联机的时间(与 /etc/shutmsg 有关)
%d	预计关闭 FTP 前已联机的中断时间(与 /etc/shutmsg 有关)

上面的表格是用在显示欢迎的讯息的，我们可以建立一个这样的档案喔！

```
[root@test root]# vi /welcome.msg
Welcome to my FTP site.
Now is the time ==> %T
The host name is %L
You are %U and from %R
There are %N person in my site, now.
If you have any problem please call me
%E

是的！内容只要上面这样即可！，马上来测试一下设定的结果

[root@test root]# ftp localhost
Connected to localhost (127.0.0.1).
220 vbird.adsldns.org FTP server (Version wu-2.6.1-20) ready.
Name (192.168.1.100:test): test
331 Password required for test.
Password:  <==输入密码
230-Welcome to my FTP site.
230-Now is the time ==> Fri Mar 21 12:03:49 2003
230-The host name is vbird.adsldns.org
230-You are vbird and from 192.168.1.100
230-There are 1 person in my site, now.
230-If you have any problem please call me
230-root@localhost
230-
230 User test logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bye
```

欢迎画面已经进入啰！ ^_^！就是这么简单即可建立一个漂漂亮亮的欢迎画面呢！甚至您也可以自行设计中文讯息，以辅助中文阅读啊！此外，您也可以在不同的目录下编辑『.message』这个

档案，则也可以在使用者进入该目录时，可以显示出该目录当中需要注意的事项喔！这些都是可以使您的 FTP 网站更佳的人性化的工具啦！

关闭 FTP 喔！

在某些特殊的情况下，可能您会要让您的 FTP 服务关闭！例如：已经预先知道时间的停电通知、已经规划好的硬件维护时间、已经设定好的系统停机时间等等，这些时候都需要关闭 FTP 。在这种情况下，通知使用者关机的时间就是一个相当好的设定啦！这个时候您就可以使用 /etc/shutmsg 这个档案啰！(注：某些情况当中，使用者可能会希望一天之中某些时候启动 FTP 而某些时刻关闭 FTP ，例如 8-16 点启动 FTP 而 16-隔天 8 点关闭 FTP ，这种情况下，您可以藉由 xinetd 这个 super daemon 的功能来达到这个目的！而不是使用这个 shutmsg 喔！切记)

我们假设这样的情况：假设我在 2003/03/30 的 12:00 要关机，并且在关机前的 60 分钟不许新的联机登入，至于已建立的联机则在关机的 30 分钟前切断联机，那么我就可以这样编辑 /etc/shutmsg 啰！(特别注意其格式！)

```
[root@test root]# vi /etc/shutmsg
2003 2 30 12 00 0100 0030
我将要关闭联机、请不要再进行联机啦！
即将关机的时间：%s
新联机失败的时间：%r
已联机关闭的时间：%d

注：上面的内容格式为两段，第一段就是时间的参数，分别是：
年 月 日 小时分钟 HHMM HHMM (最后两个分别是新联机与已联机的断线时间)
第二行以后则是一些文字的叙述！可以参考 message 里面的变量！
```

如此一来，则当 2003/03/30 的 11:00 以后，新的 FTP 要求封包将不会被联机，至于已经联机的 FTP 则在 11:30 分时，会自动的被系统踢出去喔！此外，上面这一段文字会在你登入 FTP 时显示在屏幕上面呢！可以让您很清楚的知道什么时候主机会终止 FTP 的联机啊！ ^_^！特别留意：这个档案并不会自动的消除，并且，如果设定错误，您可能无法连上 FTP ，因此，在过了那个关闭 FTP 的时间点之后，以上面的例子来说，就是 2003/03/30 ，您最好将该档案删除啊！不然可能会查不出来『到底是那个环节造成我的 FTP 无法联机』的窘境喔！ ^_^

限制最大在线人数

如果由于自己本身的硬设备问题或者是频宽的问题，所以不想提供太多的同时联机使用者数量时，可以使用『最大在线人数限制』的项目来规范呢！规范的内容很简单，同样的仅要编写 /etc/ftpaccess 即可！加入底下这一段：

```
[root@test root]# vi /etc/ftpaccess
# 规范的格式为：
# <limit> <人物群组名称> <最大联机数> <时间> <被拒绝时显示的文件档案>
```

```
limit all       20 Any          /etc/ftpmaxnumber
limit guest     10 Any          /etc/ftpmaxnumber
limit anonymous 5  Any0800-2000 /etc/ftpmaxnumber

由上面的限制当中，我们知道，在 all 这个人物群组当中，最大的同时上线
人数为 20 人，并且，这个设定是任何时刻均有效(Any)，如果你是第 21 个
联机进来我的 FTP 的人，那么您的屏幕将会出现 /etc/ftpmaxnumber 这个档案
的内容，并且『无法联机进入我的 FTP 』！请特别留意 Any 的大小写，若
写错时，这个设定将不会生效！

[root@test root]# vi /etc/ftpmaxnumber
这里已经太多人啦！请您等一下再进入！ ^_^

[root@test root]# /etc/rc.d/init.d/xinetd restart
```

这样就设定完成啦！记得重新启动 xinetd 喔！我们可以针对不同身份的使用者进行联机数目的限制呢！如同上面的设定，(请注意，那个 Any 大小写不要搞错了！另外， Any 后面没接数据，表示『任何时间』的意思，否则需要以 HHMM 的格式来书写时间的样式！)所有的联机最多只能有 20 个，至于 guest 则最多有 10 个，但是 anonymous 则在每天的 8:00 到 20:00 时限制只能有 5 个联机而已～这样就可以分别限制 OK 啦！ ^_^

限制与取消使用者的家目录规范

这个设定仅对 real users 有用啦！在早期的 Wu FTP 服务器中，由于没有考虑到一般用户可能会胡搞的问题，所以预设所有 real users 登入 FTP 主机后，可以到达任何使用者权限范围内的所有目录！例如我是 /home/test 这个使用者，那我可以进入 /etc, /var, /tmp, ... 等等目录来下载数据，无形之中对于系统的安全性有点『危险』啰！这个时候，如果我们能够将使用者『局限』在个别的家目录当中，例如 /home/test 就成为我 test 这个人的根目录 / ，由于已经被设定为根目录啦，自然也就无法去到其它的目录啰！我们可以这样做：

```
[root@test root]# vi /etc/ftpaccess

restricted-uid *
restricted-uid 200-400 test testing

[root@test root]# /etc/rc.d/init.d/xinetd restart
```

上面是两个例子喔！不要搞错了！可以分别设定，不要同时设定的啦！如果是星号『*』表示『任何身份 UID』都予以限制其家目录！如果是底下的设定，则是限制 200-400 以及 test testing 这两个使用者，让他们的家目录变成根目录，这也就是所谓的 chroot 啰！ ^_^！同时可以设定的就是限制 GID 啰！那就是 restricted-gid 啦！

那如果反过来呢？目前较新的版本(例如 Red Hat 7.3 及以后的版本)预设情况下就是设定为 restricted-uid * 的！不过，我就是要让某几个使用者可以到处浏览啊！那就使用 unrestricted-uid 以及 restricted-gid 即可！例如

```
[root@test root]# vi /etc/ftpaccess
unrestricted-uid test testing
```

可以理解吗？加油喔！ ^_^

时间相关的设定项目

或许有的时候你会发现一件事情，那就是『怎么我几分钟没有动作，我的 FTP 主机就将我踢出来咯？』就是说，当我们登入 FTP 主机后，如果隔了一段时间没有动作 (这个没有动作的时间我们叫做 idle ，也就是停顿的意思)，那么 FTP 主机会主动的认为 client 端可能使用者忘记注销了！因此会自动的将使用者踢出系统！不止如此喔！有的时候，如果您下载一个大型的档案，例如 100 MB 的数据文件好了，偏偏您是使用拨接 (理论最大下载速度 8KBytes/second)，所以应该会下载粉久粉久!但是偏偏就是 20 分钟后，FTP 主机却主动的将这个持续下载的联机关闭!这又是为什么？这是顾虑到可能有恶意者在『盗连』网站，所以才会有这样的设定!

这些时间的设定是蛮重要的啦!如果您不希望才离开一下子怎么 FTP 就踢人～那么就可以修改一下这些预设的数值啰！我们来看看底下的这些设定吧！您都可以自行修改这些设定喔！不过，请依您的需求来考虑!

```
[root@test root]# vi /etc/ftpaccess
# 时间参数预设都是以秒 为单位的!

timeout accept 120
# FTP 这个 daemon 会等待一个 PASV 的联机多久？由于 client 端的联机可能
# 受限于一些网络频宽或者其它的因素，导致无法立即连上时，我们的 daemon
# 预设就会等待 120 秒来期待 client 端的联机成功!

timeout connect 120
# 与 accept 有点类似，不过 accept 是在等待 client 端的要求之响应，而
# connect 则是在等待确认的回应！亦即三向交握内的 client 端回应 ACK 的
# 封包啰!

timeout data 2400
# 当我们下载档案的时候，最多 FTP 可以让我们下载或上传一个档案多久？
# 一般来说，由于目前大家都使用 ADSL 来传输，所以数据传送不算慢，
# 不过，万一您的档案太大了，导致传送速度很慢，那就比较麻烦啦！
# 通常这个设定的默认值是 1200 ，不过，建议可以大一点，例如 2400 可达 40
# 分钟，对于传输速度较慢的使用者会比较好一点!
```

```
timeout idle 1800
# 就是我们上面提到的，多久没有动作会被踢掉？预设是 900 ，你可以改大一点！

timeout maxidle 1800
# 与 idle 类似！不过由于 Client 端可以要求延长 idle 的时间，因此，
# 还会有所谓的 maxidle 喔！您大可将这两个咚咚都设定一样即可！

limit-time anonymous 30
limit-time guest    100
# 这个项目在设定『一次联机内，多久会被强制断线？』以上面的例子来说，
# anonymous 每次登入之后，可以取得 30 分钟的工作时间，如果超过 30 分钟，
# 系统会主动的将他踢出去！(强制断线！)这个项目对于 real user 没有效果！
```

这样就算是设定完毕啦！记得重新启动 xinetd 喔！这样子您就比较不用担心 idle 的问题啦！^_^

流量与上传下载的限制项目

关于流量上下传的限制与总体流量的限制方面， wu ftp 也提供了许多的设定来规范！我们分别来谈一谈几个常见的流量控制方法：

整体档案数目与档案容量的限额：
如果你要限制整组人员每次登入时，在该次登入可以上下传输的档案数量或容量的限额时，就需要使用到 file-limit 与 data-limit 了！可以这样使用的啦：

```
[root@test root]# vi /etc/ftpaccess
# 给予的限制情况：
# <file-limit> <in|out|total> <数目或 bytes 数> <身份群组>

file-limit out 32    alltwo
data-limit in  10240 alltwo

# 上面的设定说明是这样的， alltwo 是一个 class 群组，这个在最前头
# 规定出来的啦！而 in 代表上传、out 代表下载， total 则代表总量！
# 第一个范例是说， alltwo 这个群组当中的任何使用者，在一次登入当中，
# 可以下载的档案总数，这是以档案数量来计算的！至于第二个范例，则是说
# 在 alltwo 这个群组中的任何使用者，均仅可『上传 10KB 的容量』！
# 注意喔！那个数字代表的是 Bytes 的，要换算成 KBytes 则需要除以 1024
```

这个东西可以限制的有趣啦！不过他仅针对整个群组来进行限额的设定！这个群组是由 class 的项目来规范出来的！而且，这个规定是针对『一次登入』来规范的，也就是说，以第一个范例来说，你这次登入可以下载 32 个档案，然后就无法下载了，没关系，离线，再联机，又可以下载 32 个档案，这样应该可以理解吧？！这样也可以优化你的频宽喔！

限制流量的方法 throughput：
使用 throughput 可以限制使用者在不同的目录底下的传输速率喔！查看一下方式：

```
[root@test root]# vi /etc/ftpaccess
# 给予的限制情况：
# <throughput> <根目录> <次目录> <文件名> <bytes/s> <倍数> <地址>

throughput /var/ftp      *              *  10240  -  *
throughput /home/test    /public_html   *  51200  -  *
throughput /home/test    /realdown      *  oo     -  *.vbird.org

# 上面我设定了两个有被限制的下载目录，分别是 /var/ftp 这个目录，以及
# /home/test/public_html 这个目录，需要注意的是， 10240 代表 10Kbytes
# 喔！设定错误会让使用者下载到疯掉啊！此外，那个 /home/test/realdown
# 则是『全速』开放给使用者下载喔！
```

这个指令对于想要优化自己主机网络频宽的朋友真是太有用了!我们可以限制使用者(不论任何身份，均有效)在不同的目录底下，具有不一样的下载速度，如此一来，可以让主机的频宽花在该花的地方，而不至于被其它不明使用者占用了太多的频宽去呢！我就蛮喜欢的说～一般来说，如果您将您所提供的档案放置在一个下载目录，而这个目录里面又分为多个次目录，那么可以使用上面第二个范例的例子，将根目录与次目录分开来书写，然后可以指定不同次目录底下的传输速度，例如第二与第三个范例的样式啊！另外，特别留意的是：

- 如果想要开放全部的频宽 (就是全速、不限制下载的速度) 那就需要使用 oo 才行！注意喔！不是数字的零，而是英文字母小写的 o 喔！
- 那么什么是倍数呢？倍数就是下载的速度乘上的一个倍数值，举个例子好了，如果我的下载速度是 51200 (50KBytes)，而我想要开放到 1000KBytes ，那么那个倍数的地方就写上 20 就对了！但是，如果是 1.0 倍呢？这个时候可以写 1 也可以写『 - 』这个减号！
- 最后一个是 Client 端的主机名称或地址 (IP)。

上传、下载的比例 ratio 设定：
如果想要使用上传、下载比例的用处时，那很有可能需要重新编译你的 wu-ftp 喔！因为他需要额外的参数来加入这个功能！无论如何，我们先介绍如何使用这个设定，有兴趣的朋友可以自行研究编译的问题啊！

```
[root@test root]# vi /etc/ftpaccess
# 给予的限制情况：
# ud-dl-rate <数字> <使用者身份群组>


ul-dl-rate 2 all
```

```
# 这表示 所有的人物可以上传 1M 下载 2M 的意思!
```

请特别留意的是，这个设定项目要『Wu FTP 有支持才行』，所谓的有支持就是他必需要被编译到执行程序当中！一般来说，预设的 distribution 是有支持这个项目的，所以您可以发现这个设定是可以生效的！不过，要注意的是，由于最后面接的是『使用者群组身份』而不是 real, guest 与 anonymous 的设定，这个与 class 的相关设定有关！因此，在设定之初，最好就已经将这三组人物分别归类于三个群组当中，会比较妥当一点啦！ ^_^！也就是说，这个设定项目不能使用 real, guest 与 anonymous 等！

而如果想让使用者知道『他的上下传数据』，可以使用底下的变量功能喔！

```
%xu    可上传的 bytes 数
%xd    可下载的 bytes 数
%xR    上传与下载的比例 (1:n)
%xc    剩下的可用 bytes 数
%xT    时间限制 (分钟)
%xE    由开始登入到目前的时间预估(分钟)
%xU    上传限制 (与 file-limit 及 data-limit 的 in  有关)
%xD    下载限制 (与 file-limit 及 data-limit 的 out 有关)
```

例如编辑一个文件名称为 .message 在使用者常常下载的目录底下，让他一登入该目录就会显示该档案内容啊！内容有点像这样：

```
[root@test root]# vi .message
您可以上传/下载的比例为 1:%xR
您此次登入至目前剩下的时间：%xT
由登入到目前为止使用的时间：%xE
您可以上传的最大容量(KBytes)：%xU
您可以下载的最大容量(KBytes)：%xD
```

是不是很方便呢？！ ^_^

创造 guest user 与 guest user 的家目录问题

什么是 guest 呢？刚刚我们上面谈了这么多的信息，还是没能提到这个 guest 是什么咚咚？事实上，在 Wu FTP 当中，这个 guest 也必须要存在 /etc/passwd 当中的！他的密码也是经由比对 /etc/shadow 来达成的！那不就是『Real Users』吗？怎么会是访客？是这样的，由于担心某些使用者会使用其它的系统资源，因此，仅让这个 User 可以使用被限制住很多权限的账号，也就是说，我们将他的权限『压缩』成为访客而已，而不是让他以 real user 的身份登入我们的系统啦！

那么什么时候会用到 guest 呢？举个例子来说，如果我的主机是 WWW 服务器，而且我有开放给外部计算机（例如我的好朋友啊等等的）来使用时，那么我的朋友自然需要将网页数据传送上来对吧！然而我又不希望他会使用我主机里面其它的功能，此时，我就可以将他的账号里面关于 shell 的部分设定成为怪怪的 shell ！例如 /sbin/nologin！这样他就无法登入主机，但是还是可以进行 FTP 的联机的啦！

假如我有一个使用者账号为 test ，虽然他的家目录是 /home/test ，但是我仅想要让他可以在 /home/test/public_html 这个目录下活动，而且还可以在他进入主机给予一些讯息，此外，我也不许他可以登入系统，那该怎么作呢？

18. 修改一下该账号的 shell 部分字段，举个例子来说，假如我的账号是 test ，那么在 /etc/passwd 里面应该就会变成：

```
[root@test root]# vi /etc/passwd
.... 略 ....
test:x:511:100:testacount:/home/test:/sbin/nologin
.... 略 ....
```

19. 修改一下该使用者，让他的 shell 成为 /sbin/nologin 吧！

20. 再来将上面使用的怪怪 shell 加入到 /etc/shells 当中：

```
[root@test root]# vi /etc/shells
/bin/sh
/bin/bash
/bin/tcsh
/bin/csh
/bin/zsh
/sbin/nologin
```

21. 最底下一行是我们加入的喔！为了让 FTP 能被使用的啦！

22. 修改一下 ftpaccess 里面相关的设定啰：

```
[root@test root]# vi /etc/ftpaccess
# 几乎就是新增两行即可： guestuser 与 guest-root

guestuser  test
guest-root /home/test/public_html test

# 第一行说的是要将 test 这个使用者变成 guest 啦！
# 第二行则是说， test 这个使用者的家目录是在 /home/test/public_html
```

23.
24. 重新启动 xinetd 即可！

作一个简易的 guest user 真的不难喔！此外，这个使用者的 FTP 家目录就是 /home/test/public_html ，并且，无法离开这个目录到其它的 Linux 下的目录喔！比较安全的啦！^_^！与此同时，需要来强调的是，既然 guestuser 比较安全，那么可以将我整个系统里面的 User 都变成是 guest user ，而仅有一两个账号是 real user 吗？当然可以啰！假设我的 test1, test2 这两个是实体用户，其它的都将成为访客，那我就给他：

```
[root@test root]# vi /etc/ftpaccess

guestuser  *
realuser  test1 test2

# test1 与 test2 中间用空格隔开！这样就成功啦！
```

很简单不是吗？！

---

anonymous 的根目录与建立可上传目录

了解了 guest 的家目录设定之后，那么我们再来谈到更为麻烦的，就是那个 anonymous 啦！我们晓得在 Red Hat 7.2 当中预设的 anonymous 家目录是在 /var/ftp 当中，那么是否可以使用其它的目录来取代这个目录呢？确实是可以的，假设我们以 /home/ftp/public 做为 FTP 预设的 anonymous 家目录时，我可以加入一行设定使得这个家目录生效：

```
[root@test root]# vi /etc/ftpaccess

anonymous-root /home/ftp/public
```

不论您是否多么不愿意相信，这一行就可以让您的 anonymous 家目录的所在生效啰！不过，在预设的状态中， anonymous 是『不许上传数据』的！怎么办？没关系，我们可以设定 upload 这个参数来允许使用者上传数据到主机端呢！假设我的 anonymous 可以传送数据(档案与目录)到 /home/ftp/public/upload ，而且传送到主机的档案所属人为 ftp 所属群组则是 sys ，不过，却仅能传送档案到 /home/ftp/public/upfiles 但是不可以在这个目录当中建立其它目录！此时可以这么做：

```
[root@test root]# vi /etc/ftpaccess
# 格式很简单，就是:
# upload <家目录> <次目录> <yes|no> <档案所属人> <群组> <权限> <目录>

anonymous-root /home/ftp/public
```

```
upload /home/ftp/public /upload  yes ftp sys 0666
upload /home/ftp/public /upfiles yes ftp sys 0666 nodirs

# 第二行显示的是，我的 /home/ftp/public/upload 可以允许匿名者上传数据，
# 并且上传到主机的档案所属人与群组为 ftp/sys ，此外，档案的权限为 0666
# 至于 /home/ftp/public/upfiles 这个目录当中则仅能上传档案，不能建立目录
```

由于登入者为匿名者 (anonymous) ，所以预设是『没有身份』的，这个时候我们就必须要让上传的档案具有『身份』才行，所以才会要指定目录之外，还需要指定档案的身份啊！此外，要『真正可以让 anonymous 上传资料』还需要 Linux 档案权限的配合，举个例子来说，在 Linux 当中，我们就必须要让 /home/ftp/public/upload 这个目录可以让 ftp 使用者与 sys 群组来进行写入的工作才行！

针对人物(real, guest, anonymous)的限制设定项目

除了预设的一些设定值之外，我们还可以针对各个不同身份的使用者进行『档案权限』的控制喔！那就是跟档案权限有关的以及程序执行顺序有关的 umask 以及 nice 值啦！在设定之前，请先了解一下什么是 umask 与 nice 喔！请再次的拿出『基础学习篇』读一下里面的内容吧！底下我们就实际来介绍一下啰！

```
[root@test root]# vi /etc/ftpaccess
# nice     <数值> <使用者群组或身份>
# defumask <数值> <使用者群组或身份>

nice 10 anonymous
nice -5 real
defumask 022 real
defumask 002 anonymous

# 请务必搞懂什么是 nice 与 umask 喔！上面的 defumask 就是 default umask
# 的意思！不难理解吧？！
```

需要注意的是， nice 值只有 root 可以设定为负值，而且 nice 值越小表示这个执行程序『越快』被执行！而 umask 则是预设取消的权限，相关的说明请务必了解！(这是很重要的观念！)

拒绝某些使用者与开放某些使用者的登入

拒绝不良的 IP 或网域或 domain ：
如果您发现恶意的使用者来自于某些 IP 或者是主机名称时，而想将他抵挡住的时候，当然最好的方法还是以 iptables 将他整个踢出系统之外。但是，如果您发现的情况是，某些使用者暂时的使用方式让您的 FTP 产生了困扰，例如大量下载某些档案，或者是不当的使用 FTP 所提供的资源时，所以想暂时让他无法使用 FTP 。您想让他了解一下『为何会被取消使用 FTP 的权力』

的情况下，可以使用下列的参数来进行这样的设定信息：

```
[root@test root]# vi /etc/ftpaccess
# 准备抵挡啰！就以 <deny> <地址或主机名称> <回复给使用者讯息的文件>

deny 192.168.0.100  /etc/ftpdeny.msg
deny *.adsldns.org  /etc/ftpdeny.msg

# 上面的设定当中，当 192.168.0.100 这个 IP 来的 FTP 要求封包时，
# 不仅不提供其 FTP 的联机，并且会显示 /etc/ftpdeny.msg 这个档案的内容！
# 同样的，只要来自 .adsldns.org 的网域的计算机也都会被抵挡啊！

[root@test root]# vi /etc/ftpdeny.msg
您无法直接登入这部主机，请与您的 FTP 管理员联络！
管理员 E-Mail 地址为： %E
```

如此一来，则当 192.168.0.100 这个来源 IP 的任何使用者来连接 FTP 时，在他们的 FTP 软件上面，就会显示出问题的所在啦！至于联络人则是以 %E 这个 email 变数做为联系的！

拒绝某些危险的账号：
在我们的 Linux 系统上面，由于主机套件上面的需求，所以都会预设有一些基本的账号存在的！例如常见的 adm, bin, sys, 以及 mail 上面常见的 mail ！同时，也有一些常见的群组存在的！要注意的是，这些账号通常仅有系统工作的时候才会用到，其它一般身份使用者不太会使用这些账号的啦！此外，由于系统账号常常会局限在 UID 1~499 以内，而大于 65000 以上的账号应该不常见的！所以，我们可以拒绝使用这些 UID 与 GID 来登入 FTP 主机的！这就需要用到 deny-uid 与 deny-gid 的设定项目啦！此外，事实上，如果您也发现某些可疑的账号时，也可以先以这个设定项目将他拒绝在您的系统之外喔！使用的方法如下：

```
[root@test root]# vi /etc/ftpaccess
# 就直接使用最初的设定 deny-uid <账号、UID 或范围>

deny-uid %-499 %65000-
deny-gid %-499 %65000-
allow-uid ftp
allow-gid ftp
```

这样的方式也可以用来抵挡一些不合法的 UID 啊！

使用额外档案来抵挡：/etc/ftphosts
我们也可以使用 /etc/ftphosts 来进行使用者仅可以用来联机的主机喔！这个档案的设定方法有点像底下这样：

```
[root@test root]# vi /etc/ftphosts
```

```
# 格式为 <deny> <使用者账号> <不许联机的 IP 或主机名称>
# 格式为<allow> <使用者账号> <不许联机的 IP 或主机名称>

deny  test    192.168.0.0/24
allow testing 192.168.1.0:255.255.255.0
# 注意上面这两种书写方式的不同的!

deny test2    192.168.5.10 allow test2 *
```

上面说明的是，我不许 test 由 192.168.0.0/24 这个网域来！但是允许 testing 这个使用者来自 192.168.1.0/24 ！请注意这两种方式！当以 bit 书写时，就直接加上 / ，至于如果是以 netmask 来书写时，则是加上冒号『:』。需要特别注意的是，如果同时存在两条同一个人的设定，例如：

```
deny  test *
allow test *
```

这样的情况下『最后出现的那一条规则为预设的规则！』所以结果就是 test 可以由任何地方进来 FTP 主机喔！这里还请特别留意啊！此外，进行完上面的设定后，请记得重新启动 xinetd 啊！

使用 PAM 模块的机制来抵挡：

上面的几个案例都是在 FTP 的设定档里面规定的，也就是说，上面的设定都是您的 Client 端已经进入在 FTP 里面之后再来进行抵挡的动作的！那么有没有还不需要动到 FTP 就抵挡的机制呢？除了我们后面才会提到的 iptables 以及 TCP_Wrappers 之外，我们还有个 PAM 密码验证模块可以利用呢！首先来看一下 PAM 模块里面关于 FTP 的基本内容：

```
[root@test root]# vi /etc/pam.d/ftp
#%PAM-1.0
auth       required     /lib/security/pam_listfile.so item=user sense=deny
file=/etc/ftpusers onerr=succeed <==此行与上行为同一行
auth       required     /lib/security/pam_stack.so service=system-auth
auth       required     /lib/security/pam_shells.so
account    required     /lib/security/pam_stack.so service=system-auth
session    required     /lib/security/pam_stack.so service=system-auth
```

仔细的看一下上面的档案喔！粗体的部分那一行里面的 sense 是 deny 喔！这表示写到 /etc/ftpusers 里面的使用者名称代表『无法登入 FTP 系统的使用者』！哈哈！所以说，只要是这个档案里面的任何使用者就无法通过密码验证 (由 PAM 模块管理的) 这一关，自然也就无法使用 FTP 了！因此，当您不想让某个 User 进入你的 FTP 使用者，只要将他的名字写入这个 /etc/ftpusers 当中，那该使用者就无法使用 FTP 了！但是不会影响到其它的 daemons 的服务喔！还不错吧！ ^_^

让 root 可以使用 FTP 功能：

除了上面提到的之外， /etc/ftpusers 里面的内容大部分的账号都是来自于系统账号，这包括每个 Linux 都会存在的 root 这个账号！所以 root 自然也就无法登入系统的 FTP 服务了。如果我真的想要让 root 可以使用 FTP 的功能，我需要怎么作呢？

25. 首先，请先将 /etc/ftpusers 里面的 root 拿掉；
26. 确认 /etc/ftphosts 没有挡掉欲登入的主机的 IP 或者是 hostname；
27. 确认 /etc/ftpaccess 里头关于 deny-uid 与 deny-gid 没有 0 这个数字存在！
28. 重新启动 /etc/rc.d/init.d/xinetd 这个 daemon ！

这样大概就可以让 root 登入啦！不过『强烈建议不要这么做！』

目录与连结文件的问题

除了使用连结档之外，我们可以直接让使用者以『 cd {某目录名称} 』进入到该目录下！而且是不论在任何目录下均可进行这个动作喔！这样一来有个好处啦，就是如果很多 real user 的档案当是集中在某个目录底下的，那么直接使用这个功能就不需要到每个使用者的家目录里面建立连结文件啰！

```
[root@test root]# vi /etc/ftpaccess
# alias <你想要给这个目录起的名字> <实际 Linux 系统的目录>

alias  anonymousdir  /var/ftp
```

如同上面的设定，当使用者在任何地方只要下达『 cd anonymousdir 』就可以进入到 /var/ftp 这个目录当中啦！请注意，这个 alias 『仅针对 cd 这个指令有用』！您可以将这个 anonymousdir 目录定义写在 /welcome.msg 当中，让使用者可以知道有什么功能呢！还不错吧！但是有个问题必须要了解的，那就是我们知道了 restricted-uid 可以限制住某个使用者让他仅能在『家目录』当中活动，而无法移动到其它目录去！现在想象一个案例，如果我的 test 这个使用者被限制在家目录 /home/test 里面，那么如果我在 /home/test 里面建立一个连结档，连结到 /home/ftp 这个目录，那么是否 test 就可以利用这个连结档移动到 /home/ftp 去呢？还有，这个 alias 所建立的目录是否可以让使用者离开自己的家目录呢？『很抱歉』答案是否定的！也就是说，不论是建立连结档或者使用 alias ，您都无法离开被限制住的目录喔！

此外，为了避免 anonymous 这些匿名者上传的时候传送一些奇怪的档名，例如传送 windows 里面的文件名称，那就有可能包含了很多怪异的特殊字符，这些特殊字符可能会造成我们 Linux 系统的一些困扰。为了避免这些问题，我们可以『指定』anonymous 不能上传某些怪异的文件名称的档案，使用下面的过滤机制：

```
[root@test root]# vi /etc/ftpaccess
# path-filter <群组> <讯息档案> <允许字符> <不许字符 1> <不许字符 2> ...

paht-filter  anonymous  /etc/pathmsg  ^[-A-Za-Z0-9._]*$  ^\.  ^-

[root@test root]# vi /etc/pathmsg
请注意，您的文件名称不符合本站的限制，请检查：
1. 文件名起始字符需为英文或数字或底线；
```

```
2. 文件名起始字符不可为 . 或者减号 -
```

在上面的设定当中，我们可以发现，在 /etc/pathmsg 后面除了第一个是『允许』的字符外，其它后续接的都是『不允许的字符！』至于这些字符的规格则与正规表示法『Regular Expression』有关喔！上面的意思是说，档名开头(^符号表示开头)只能是英数字，而不能是小数点与减号！这里请特别注意正规表示法喔！他是很重要的！请再次的劳驾到『基础学习篇』察看正规表示法吧！而万一使用者上传的档案档名是『错误的，主机不允许的』情况时，则会将 /etc/pathmsg 这个档案里面的讯息告知使用者！当然，这个档名是可以变动的！

建立 passive port 提供 client 端登入

建立 passive port 给 client 端使用是有其必要性的！这个我们在前言的部分已经提过了！但是，又由于可能会导致『难以追踪入侵者』的问题，让这个 passive 变的棘手～此时， passive ports 这个可以指定 port number 的设定就可以让我们定义出少量的 port 来做为 passive 之用啰！那该怎么作呢？更加的简单！只要几个设定就可以搞定啦！

```
[root@test root]# vi /etc/ftpaccess
# passive ports <CIDR 地址> <最小 port> <最大 port>
# pasv-allow    <人员身份> <地址>

passive ports 0.0.0.0/0 65501 65505
pasv-allow all *

# 这代表来自任何地方的 IP 在要求 passive 联机模式时，将以 65501 ~ 65505
# 之间的 port 来做为他们 PASV 联机的要求啦！如此一来，则 passive ports
# 将仅会随机选取 65501 ~ 65505 之间的 5 个 port 来做为 PASV 之用，
# 其它的 port 将不会使用到 FTP 的 passive 模式！如此一来还可以建置
# 防火墙上面的 port mapping 呢！很不错吧！ ^_^
```

就这样一个设定就可以让您的 passive 模式的联机当中，限制他的相关的 ports 啰！此外，如果您的 FTP 是在 NAT 内部的话，那么就可以就由 firewall 的 port mapping 的动作来进行 PASV 的转 port 作用来达成内部 FTP 主机的被动联机喔！ ^_^

修改 FTP 预设的 port 21 的联机

在约略了解了大部分常用的 Wu FTP 功能之后，咦！如果我不想使用 21 这个 port 做为我 FTP 预设的指令信道，那么是否有办法修改这个 port 号码呢？举个例子来说，我希望我的 FTP port 为 3366，有的！就是利用两个档案，分别是：

- Xinetd 这个 Super daemon 的设定档：/etc/xinetd.d/wu-ftpd
- Port 对应 daemon 名称的档案：/etc/services

修改的方法很简单！我们先来看一下 wu-ftpd 这个档案的内容之后，再来进行进一步的说明：

```
[root@test root]# vi /etc/xinetd.d/wu-ftpd
service ftp
{
        disable = no
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/in.ftpd
        server_args             = -l -a
        log_on_success          += DURATION USERID
        log_on_failure          += USERID
        nice                    = 10
}
```

事实上，重要的地方就是那个 service ftp 的地方！这个 ftp 的 port 就是写在 /etc/services 当中的啦！所以说，如果我将 /etc/services 里面的 ftp port 修改一下，那就可以改 port number 啦！不过，这还不是个好主意～我还可以透过自行给予的 daemon name 来进行设定喔！举个例子来说，如果我要建立一个名为 vbftp 的 daemon 名称，那我可以这样做：

```
1. 修改 wu-ftpd 这个里面关于 daemon 的名称：

[root@test root]# vi /etc/xinetd.d/wu-ftpd
service vbftp <==修改这里就对啦！
{
        disable = no
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/in.ftpd
        server_args             = -l -a
        log_on_success          += DURATION USERID
        log_on_failure          += USERID
        nice                    = 10
}

2. 修改 /etc/services 的 daemon 相关设定内容
[root@test root]# vi /etc/services

vbftp        3366/tcp
```

```
# 上面这一行可以加在这个 /etc/services 里面的最后一行！自己设定的啦！

3. 重新启动 xinetd

[root@test root]# /etc/rc.d/init.d/xinetd restart
```

这样就可以修改您的预设 FTP port 啰！加油是看看！

一个多样化的实例

好了，大致上 Wu FTP 的设定您应该可以理解了吧！那么现在出个例题给大家思考一下，假如我需要达成底下的规范，那么应该如何设定 Wu FTP 相关的档案呢？

31. 将群组设定成为 real, guest, anonymous 这三个群组分别控制！(用 class 来区分喔！)；
32. 用户身份有 real, guest, anonymous，其中， real 仅允许来自 140.116.0.0/16 这个 B class 的网域，其它两者虽可来自于所有网域，但不允许来自 61.141.0.0/16 这个 B Class 的网域；
33. 允许使用 passive ports ，port number 设定为 65501 - 65510 这 10 个 ports ；
34. 系统里面小于 499 以及大于 65000 的 UID 与 GID 都被拒绝登入；
35. 任何时刻，最大在线人数限制为 30 人，其中， guest 最多 10 人，而 anonymous 最多为 5 人；
36. 实体用户 mysiteuser 被限制他仅能在他的家目录当中工作，无法离开其家目录，至于其它实体用户则不在此限；
37. 我有两个 WWW 的个人用户 wwwuser1 与 wwwuser2 ，我将这两个用户订为 guest 身份，为使他的网页数据传输方便，将他的 FTP 主机的家目录设定为他的 WWW 家目录，亦即为 /home/wwwuser1/public_html ，并且设定传输速度最大为 100 Kbytes/second。注意，我的 ftp 使用者预设的群组为自行建立的 myftpusers；
38. 我有一个 FTP 交换连结用户 ftpuser，我与他交换连结，连上下传都需要注意！上下传比例为 1.0 ，并且限制他传输的速度为 64 Kbytes/second ；
39. 其它的匿名登入者的家目录设定为 /var/ftp 这个目录当中，并且限制 anonymous 一次联机最久 10 分钟，而且最多仅能下载 20 个文件，以及 10MB 的数据量，此外，传输速度最快仅能到达 32 Kbytes/second。至于上传的设定方面，仅允许上传到 /var/ftp/upload 这个档案，此外，预设档案拥有者为 ftp 群组是 sys，上传速度最快为 16 Kbytes/second；

似乎很麻烦，那我们就来一个一个的设定吧！

40. 先设定使用者的账号：
    上面的案例中，我们共有 wwwuser1, wwwuser2, ftpuser 与 mysiteuser 这四个人，由

于 mysiteuser 为实体用户，所以不要去改他的登入 shell ，至于 wwwuser1, wwwuser2 与 ftpuser 由于他的群组需建立成为 myftpusers ，所以需要设定一下啦！假设 myftpusers 尚未建立，所以得先建立群组才能建立使用者！并且这两个使用者单纯的想要使用 FTP 而已，所以直接加上 shell 为 /sbin/nologin：

```
[root@test root]# groupadd myftpusers
[root@test root]# useradd -m -g myftpusers -s /sbin/nologin wwwuser1
[root@test root]# useradd -m -g myftpusers -s /sbin/nologin wwwuser2
[root@test root]# useradd -m -g myftpusers -s /sbin/nologin ftpuser
[root@test root]# passwd wwwuser1  <==不要忘记给三个人密码！
```

41.
42. 开始设定我们的 /etc/ftpaccess 内容：
    设定的项目我们将他分为(1)主机设定(2)实体用户设定(3)访客(4)匿名者等部分来设定啰：

```
[root@test root]# vi /etc/ftpaccess
# 底下的数据为 VBird 的 FTP 主机设定范例
#
########################################################################
# 1.  针对 Server 的设定项目：
########################################################################
# 1.1 针对群组的设定项目
class          all            real,guest,anonymous    140.116.0.0/16
class          allreal        real                    140.116.0.0/16
class          allguest       guest                   !61.141.0.0/16  *
class          allanonymous   anonymous               !61.141.0.0/16  *

# 1.2 其它主机相关的设定项目
email          vbird@tsai.adsldns.org
hostname       tsai.adsldns.org
shutdown       /etc/shutmsg
loginfails     3
log            transfers      anonymous,guest,real    inbound,outbound
passwd-check   rfc822         warn

# 1.3 讯息管理
readme         README*        login
readme         README*        cwd=*
message        /welcome.msg   login
message        .message       cwd=*

# 1.4 指令管理
compress       yes            all
```

```
tar             yes             all
chmod           no              guest,anonymous
delete          no              anonymous
overwrite       no              anonymous
rename          no              anonymous

# 1.5 人物登入管理
deny-uid        %-499           %65000-
deny-gid        %-499           %65000-
allow-gid       myftpusers

# 1.6 时间相关的设定值
timeout data    2400
timeout idle    1800
timeout maxidle 1800

# 1.7 主机最大联机人数设定
limit   all             30      Any     /etc/ftpmaxnumber

# 1.8 被动的 port 设定
passive ports   0.0.0.0/0       65501   65510


########################################################################
# 2.  针对实体用户的设定
########################################################################
restricted-uid  mysiteuser


########################################################################
# 3.  针对 guest 用户的设定
########################################################################
limit           allguest        10      Any     /etc/ftpmaxnumber
guestuser       wwwuser1        wwwuser2        ftpuser

# 3.1 www users
guest-root      /home/wwwuser1/public_html      wwwuser1
guest-root      /home/wwwuser2/public_html      wwwuser2
throughput      /home/wwwuser1/public_html      *   *   102400  -       *
throughput      /home/wwwuser2/public_html      *   *   102400  -       *

# 3.2 FTP users
guest-root      /home/ftpuser                   ftpuser
ul-dl-rate      1       allguest
throughput      /home/ftpuser   *       *       64000   -       *
```

```
########################################################################
# 4.  针对 anonymous 用户的设定
########################################################################
limit           allanonymous    5       Any     /etc/ftpmaxnumber
anonymous-root  /var/ftp
limit-time      anonymous       10
file-limit      out             20              allanonymous
data-limit      out             10000000        allanonymous
throughput      /var/ftp        *       *       32000   -       *
throughput      /var/ftp        /upload *       16000   -       *
upload          /var/ftp/upload yes ftp sys 0666
```

43.

44. 建立可上传目录与使用者家目录：

由于我们设定了 /var/ftp/upload 为可上传的目录，所以需要动手设定一下啰：

```
[root@test root]# mkdir /var/ftp/upload
[root@test root]# chown ftp:sys /var/ftp/upload
[root@test root]# mkdir /home/wwwuser1/public_html
[root@test root]# mkdir /home/wwwuser2/public_html
[root@tset root]# chown wwwuser1:myftpusers /home/wwwuser1/public_html
[root@tset root]# chown wwwuser2:myftpusers /home/wwwuser2/public_html
```

45.

46. 修订一下 /etc/shells：

必须要确认 /sbin/nologin 在这个档案内

```
[root@test root]# vi /etc/shells
/bin/bash
....(略)...
/sbin/nologin
```

47.

48. 修改一下欢迎画面档案：

你可以建立一下欢迎画面喔！档案有这些：

/welcome.msg
/home/wwwuser1/public_html/.message
/home/wwwuser2/public_html/.message
/var/ftp/welcome.msg

如果想要一劳永逸，那么将 /welcome.msg 这个档案复制到 /etc/skel/welcome.msg 以及 /etc/skel/public_html/.message 则是一个不错的主意！

49. 重新启动 xinetd 啰！

大致的流程就是这样啦！ ^_^

Client 端的使用 FTP 软件：

事实上，我们在网络常用指令那个章节当中已经介绍过了 ftp 与 ncftp 这两个很好用的 client 端软件了，在这里我们再次的强调一下这两个软件，其中，比较重要的是强调传输的模式。在 Server 与 Client 传输的过程中，数据的流动主要分为 binary 与 ascii 两种模式，需要注意的是：

- Binary 的传送方式当中，FTP Server 并不会去改变档案的内容，所以数据得以完整的呈现；
- 但在 ASCII 传输模式当中，主要将数据视为一般的纯文字文件，例如：原始码或者是设定档等等，在这种传输模式当中，Server 会将档案以一行一行来传送，所以如果您以 ASCII 传送经过编译过的 binary program 时，将可能导致无法执行的问题(因为被转成文字文件啦！)

底下我们就来介绍两个软件吧！

- ftp

远程传送数据当中，速度最快的协议之一

语法：

```
[root @test /root]# ftp [-p] host [port]
参数说明：
-p ：启动 PASSIVE 模式！
范例：
[root @test /root]# ftp localhost          <==预设是以 port 21 来进行联机
[root @test /root]# ftp localhost 1354
如果你设定的 ftp 的 port 非正规的 21 ，则可以这样！

[root @test /root]# ftp localhost    <==连接到远程主机
Connected to localhost (127.0.0.1).
220 localhost FTP server (Version wu-2.6.1-20) ready.
Name (127.0.0.1:test): test
331 Password required for test.
Password: <==输入密码
230 User test logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir                    <==显示远方主机的内容
ftp> cd                     <==变换远程主机的目录
```

```
ftp> close or bye or exit  <==离开远程主机
ftp> get file              <==取得远程主机的档案
ftp> mget file             <==取得所有的档案，例如 mget .bash* !
ftp> put file              <==将本地端档案 file 丢到远程主机上
ftp> mput file             <==与 mget 差不多意思啦!
ftp> delete file           <==杀掉远程主机的 file 档案
ftp> help                  <==显示求救指令!
ftp> mkdir dir             <==在远程主机上面建立目录
ftp> lcd                   <==变换本地端路径!
ftp> ascii or binary       <==重要的很! 上面提过的!
```

- ncftp

登入匿名主机最好用的文字接口 FTP 软件!

语法：

```
[root @test /root]# ncftp [host]
[root @test /root]# ncftp [ftp://domain.name/path]
参数说明：
可以直接连接到 host 主机，也可以直接连接到主机的某个路径之下，相当的方便
范例：
[root @test /root]# ncftp ftp.nsysu.edu.tw     <==连接到中山大学 FTP 站
[root @test /root]# ncftp ftp://ftp.nsysu.edu.tw/Linux
直接连接到 Linux 底下的 Linux 目录下!

远程主机的一些服务指令：
ncftp /pub/Linux > cd Redhat             <==变换目录
ncftp /pub/Linux > dir                   <==显示目前目录下的档案与目录信息
ncftp /pub/Linux > get file1             <==将 file1 的资料存到本地端
ncftp /pub/Linux > get -z file1 file2    <==将 file1 存到本地改档名为 fil12
ncftp /pub/Linux > get -A file1 file2
将 file1 以累积的方式(append)增加到 file2 这个档案
ncftp /pub/Linux > pub file              <==将档案由本地端上传至远程!
ncftp /pub/Linux > rename file1 file2    <==将远程的主机之 file1 更名为 file2
ncftp /pub/Linux > rm file               <==删除档案
ncftp /pub/Linux > rmdir directory       <==删除目录
ncftp /pub/Linux > mget directory    <==可以下载『整个目录』的数据! 很棒吧!

本地端主机的一些指令功能：
ncftp /pub/Linux > lcd          <==变更本地端目前所在的目录
ncftp /pub/Linux > lls          <==显示目前本地端所在目录的档案与目录信息
ncftp /pub/Linux > lmkdir       <==在本地端建立目录
ncftp /pub/Linux > lpwd         <==显示目前本地端主机的所在目录
ncftp /pub/Linux > lrm          <==删除本地端的档案
```

```
ncftp /pub/Linux > lrmdir        <==删除本地端的『目录』
```

Server 端的安全设定项目：

事实上，由近年来公布的安全漏洞信息来看，这个 Wu FTP 所造成的漏洞真的是不少，所以，在架设 Wu FTP 时，一定要注意到安全性的设定项目上面，这个真的很重要！因为由网络上朋友回报的信息来看，目前最大宗的 Linux distribution (7.x 版本) 连上 Internet 之后，被入侵的最热门 port 就是 Wu FTP 啰！所以，我们才会一再地强调『非必要，不要架设 FTP 』。一般来说，我们会这样建议大家：

1. 除非真的必要，否则尽量将 FTP 的 port 21 关掉；
2. 如果真的要架设 Wu FTP ，请『一定要』将 Wu FTP 更新到最新的版本；
3. 更新完毕之后，在架设之前，一定要考虑清楚，是否要开放 anonymous 的联机？除非必要，否则仅开放 Real 以及 guest 使用者登入即可；
4. 设定时，最好在 /etc/ftpaccess 就限制联机者的地址，以杜绝可能的入侵者的尝试；
5. 尽量将使用者限制在他们自己的家目录内；
6. 将不想让他联机的账号以 PAM 模块来抵挡，亦即将使用者账号写入 /etc/ftpusers 里面即可；
7. 可以的话，以 iptables 以及 TCP_Wrappers 架设两层防火墙，限制住开放的网域即可。

由设定、PAM 模块、TCP_Wrappers、iptables 共有四层防火墙，这样会比较安全一些，不能说就『高枕无忧』啦，但是会比较放心一些！我们约略的提一下防火墙的相关设定吧！

---

iptalbes

最外层的防火墙可以说就是这样 iptables 了，在设定之前，我们就必需要了解的是，我的 port 是几号？一般来说， port 至少有 21 以及 20 两个，但是不要忘记了，还有 passive ports 喔！这里我们假设 FTP 仅允许来自 140.116.0.0/16 这个 B Class ，而 passive ports (假设我们没有使用 related 那个防火墙设定项目)开放 65501 至 65505 共五个，那么我就应该在我的防火墙规则加入这几段：

```
/sbin/iptables -A INPUT -p TCP -i eth0 -s 140.116.0.0/16 --dport 20:21    \
    -j ACCEPT
/sbin/iptables -A INPUT -p TCP -i eth0 -s 140.116.0.0/16 --dport 65501:65505\
    -j ACCEPT
```

这样就限制了 FTP 仅可以使用的 ports ，其它更多的防火墙规划，请参考『简易防火墙』那个章节吧！

TCP_Wrappers

TCP_Wrappers 的设定我们在『简易防火墙』那一章节已经提过了，这里仅针对 FTP 提出设定项目：

```
[root@test /root]# vi /etc/hosts.allow
in.ftpd : 140.116.0.0/255.255.0.0

[root@test /root]# vi /etc/hosts.deny
in.ftpd : ALL : spawn (/bin/echo Security notice from host `/bin/hostname`; \
/bin/echo; /usr/sbin/safe_finger @%h ) | \
/bin/mail -s "%d -%h security" root@localhost & \
: twist ( /bin/echo -e "\n\nWARNING connectin not allowed. Your attempt has been logged. \n\n\n 警告
您尚未允许登入，您的联机将会被纪录，并且作为以后的参考\n\n ". )
```

这样一个小型的防火墙就建置起来啦！而且，只要有人 scan 你主机的 ftp port ，就会立刻被记录下来喔！

---

pam 模块与 /etc/ftpusers 的关系

要登入 FTP 之前需要输入密码对吧！输入密码就可以利用 PAM 模块来进行过滤啰！目前我们使用的 PAM 模块是放置在 /etc/pam.d 这个目录中，而 ftp 则是 /etc/pam.d/ftp 这个档案，在前头我们也约略提过这个档案啦，主要就是将要被抵挡的使用者账号写入到 /etc/ftpusers ，那么该账号想要登入 FTP 时,就会被 PAM 模块的过滤系统挡掉!相当简易吧!而且还很有效呢！^_^

FTP 本身提供的抵挡 username 或 host 的控件目

除此之外， FTP 的设定档里面 /etc/ftpaccess 也有 deny 与 deny-uid 等等的抵挡机制，您也可以使用 /etc/ftphosts 来抵挡,关于这些档案的设定在本章前面都提过了,请翻阅参考啰！^_^

重点回顾

- FTP 是 File Transfer Protocol 的简写，主要的功能是进行 Server 与 Client 端的档案管理、传输等事项；
- Wu FTP 是很常见的一个 FTP 软件，不过由于安全性，建议务必升级到最新版本，此外，他的主要设定档是 /etc/ftpaccess 这一个；
- 除了 Wu FTP 这个 FTP 软件之外，其实可以使用 SSH 提供的 sftp 功能来取代 FTP；
- FTP 这个 daemon 比较常以 super daemon 来管理，亦即 xinetd 或者是 inet 这两个 super daemon ；
- FTP 这个 daemon 所开启的正规的 port 为 20 与 21 ，其中， 21 为指令信道， 20 为数据传输信道；
- FTP 的传输路线主要分为主动与被动(Passive, PASV)，如果是主动的话，则 ftp-data 以 20 传送，否则则以 /etc/ftpaccess 规定的 passive ports 或者随机选取大于 1024 的 port 来进行被动式联机模式；

- 一般来说， FTP 上面共有三个群组，分别是实体用户、访客与匿名登入者(real, guest, anonymous)；
- 可以藉由修改 /etc/passwd 里面的 Shell 字段，来让使用者仅能使用 FTP 而无法登入主机；
- 可以使用 guestuser 及 guest-root 来限制实体用户，使成为访客用户；
- 需要以 upload 来设定可以让 anonymous 上传的目录，并设定好其权限喔！
- FTP 的指令、与使用者活动所造成的登录档是放置在 /var/log/xferlog 里面；
- 在 Client 端使用 ftp 这个程序时，可以加上『 ftp -p hostname』来让联机变成 passive 模式。

参考资源

- man ftpaccess
- Study Area ：http://www.study-area.org
- Wu FTP 官方网站：http://www.wu-ftpd.org

本章习题练习 (要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看)

- FTP 在建立联机以及数据传输时，会建立哪些联机？
- FTP 主动式与被动式联机有何不同？
- 有哪些动作可以让您的 FTP 主机更为安全 (secure) ？
- 我们知道 ftp 会启用两个 ports ，请问这两个 port 在哪里规范的？而且，一般正规的 port 是几号？
- Wu FTP 的主要设定档在哪里？
- 在 Wu FTP 的设定档当中，那个 log transfer 是干嘛用的？
- 在 Wu FTP 的设定档当中，那个 passive ports 是干嘛用的？
- 那一个档案可以用来抵挡类似 root 这种系统账号的登入 FTP？
- 在 FTP 的 server 与 client 端进行数据传输时，有哪两种模式？为何这两种模式影响数据的传输很重要？

简易 FTP Server 架设 -- Pro FTPD

最近更新日期：2003/09/05

我们知道在 Internet 上面有个很快速的档案传输协议，就是 FTP ！而且也知道最古老的 FTP 服务器软件之一就是那个很出名的 Wu FTP 啰！但是，虽然 Wu FTP 的速度快、架设方便，不过由于招牌老且大，所以『深受怪客 （ Cracker ） 的喜好』啊！导致 Wu FTP 的安全性堪虑～此外， Wu FTP 受限于他的架构问题，所以一些在 Win32 上面执行的 FTP 功能 （ 例如很出名的 Server-U ） 在 Wu FTP 上面都没有办法很简易的就达成这样的功用！为了改善安全上面的疑虑以及增强 FTP 软件的设定便利性，所以就有这个 Professional FTP daemon (proftpd) 的产生啦！这个 proftpd 并非用来与 Wu FTP 打对台的，但是由于他的设定弹性太高了！所以渐渐的大家都倾向于使用这个 FTP 软件来架设自己的 Linux 服务器呢！呵呵！如果你是使用 Wu FTP 的使用者，也可以尝试以这个 proftpd 来取代 Wu FTP 喔！应该会更安全的啦！而，如果您习惯使用 Server-U 来设定特殊账号的上传/下载数据的话，呵呵！那么 proftpd 就是您转换跑道的首选了！

前言：

众所皆知的， FTP 是一个行之有年的网络通讯协议，我们可以透过 FTP 这个协议在不同的作业平台上面进行档案的传输、删除与移动等等的工作，而使用最为广泛的 FTP 架设软件就是那个 Wu FTP 了！但是由于 Wu FTP 毕竟在『安全历史』的过程中，实在是被发现了太多的危险漏洞了，所以目前有相当多强调安全性的 FTP 服务器软件渐渐抬头，其中之一就是这个有名的 Professional FTPD 了！底下我们来谈一谈为何需要有这个服务器软件以及相关的其它说明吧！

为什么要使用 Professional FTP daemon 呢？

既然 Wu FTP 这个服务器软件并不是十分的安全，所以这个 Pro FTPD 当然主要就是以较为安全的角度去设计的一个全新的 FTP 服务器软件了！在 ProFTPD 的官方网站上面也提出了，最早设计这个 FTP 的理念不是想要『干掉 Wu FTP 』，而是希望给予大家一个更为安全，且在设定上面更为便利的一个 FTP 服务器软件啰！除了安全性之外，为何还要强调『设定便利性』呢？这是因

为目前在 Windows 的系统当中 (Win32)，有个相当有名气的 FTP 服务器软件，那就是鼎鼎大名的 Server-U 啰！这个 Server U 实在是很厉害，在设定上面相当的简易，此外，还可以根据不同的使用者给予不同的传输速度与上传、下载比例，设定上面又很有弹性，实在是难能可贵的一套软件！那么我们的 Wu FTP 能否达到这样的功能呢？当然可以啰！不过.....设定上确实比较麻烦～因此上，这个 proftpd 可就帮了个大忙啰！

基本上， ProFTPD 主要具有底下的几个特征：

- 主要的设定档仅有一个，设定上甚为简易；
- 每一个开放出去的目录底下的 .ftpaccess 可以用来取代 ProFTPD 的主要设定档规范的参数， .ftpaccess 功能类似于 Apache 的 .htaccess 喔；
- 设定 FTP 成为具有虚拟 FTP 主机与匿名登入 FTP 主机的设定甚为简易；
- 可以依据个人的设定要求，以 stand-alone 的方式或者 inet/xinet (Super daemon) 的管理方式来启动；
- 匿名登入时，使用者所登入的目录底下，不需要额外的 binary 执行程序的支持，具有较佳的安全性；
- 不需要 Linux System 本机的执行程序的支持，由于使用 Linux 本机的程序可能会造成系统安全上的顾虑， ProFTPD 在自己的原始码当中已经含有所需要的执行指令了，所以不需要系统的 binary 执行文件的支持，系统安全上面较可靠；
- 仍然具有 Linux 系统本身的 user/owner 权限属性，以及隐藏文件的属性等等均存在；
- 使用者登入 ProFTPD 时，登入的信息将会存放一份在 utmp/wtmp 的登录档中，这是什么呢？这就是使用 last 可以显示出登入信息的重要登录文件啰！
- 登录密码可支持 Shadow 密码档案 (/etc/shadow)，亦同时支持已经死亡的账号 (请参考『鸟哥的 Linux 私房菜 -- 基础学习篇』里面的 账号管理部分关于 shadow 的介绍)

看起来觉得真的很不错吧！呵呵！尤其是那个 .ftpaccess 档案更是能够引起使用者的『兴趣』呢！怎么说呢？还记得在 Apache 里面如果使用者想要架设一个属于自己的个人首页时，可以依照 .htaccess 设计自己的风格啊！同样的，在 ProFTPD 当中，使用者也可以藉由 .ftpaccess 这个档案来『设计属于自己的 FTP 主机』喔！可以不必依照 Linux 本机 FTP 服务器软件的僵化设定呢！呵呵！真是粉不错喔！

架设之前你需要了解的原理

与之前我们所说明各个 Server 的架设时需要知道的原理一样，这里我们还是得针对 FTP 的『联机原理』来说明一下，但是这部份我们已经在 Wu FTP 里面说过了，所以请前往 Wu FTP 那一章节好好阅读一下吧！在开始 FTP 的设定之前，你必须要知道的原理有：

- FTP 在 Client 与 Server 进行联机时，主要使用到的 port 有几个？分别具有什么用途？

- Client 与 Server 进行 FTP 联机时，其模式分为 Active 与 Passive ，这两种模式的差异为何？
- 当 FTP 架设在防火墙内部时，则这个 FTP 需要使用的联机模式为何？

如果你能够了解上面的几个细节，那么设定 FTP 是一点也不困难的呢！底下我们就来谈一谈这个好用的 proftpd 吧！

套件安装：

虽然 Wu FTP 可能还是目前使用上最广泛的 FTP 服务器软件，不过，毕竟安全上面有点小问题，所以近年来各主要的 Linux distributions 在发布 FTP 服务器软件时，渐渐的都以 ProFTPD 来取代 Wu FTP 了！例如近期的 Mandrake 9.x 等等！而由于这些主要的 distribution 使用的是 RPM 的安装方式，因此，呵呵！如果您的系统是属于这些较近期的版本，那么就直接以 RPM 的方式来安装这个 FTP 服务器软件即可！

不过，毕竟使用 Wu FTP 的朋友大有人在，而且在提供 Wu FTP 套件的 distributions 通常是不提供 proftpd 的 RPM 版本的，所以这里我们主要也以 Tarball 的方式来安装 ProFTPD 。如此一来，不论您原先是使用 Wu FTP 还是原本就是 ProFTPD ，都可以经由 Tarball 的方式来重新安装一次你的 FTP 服务器软件呢！废话不再多说了，赶紧来看看怎么以 Tarball 的方式安装 ProFTPD 吧！

- 下载 proftpd：
  您可以前往 ProFTPD 的官方网站下载 proftpd ，不过，蛮建议在台湾的中山大学 FTP 网站下载的，他的速度也是粉快的喔！中山大学关于 ProFTPD 的网址在：http://ftp.nsysu.edu.tw/Unix/FTP/proftpd/distrib/source/，我在这里测试的版本是 1.2.8 这个在 2003/03 出的最新版的 ProFTPD 喔！所以他的档名应该是：proftpd-1.2.8.tar.gz 这个档案的啦！你也可以在我们网站下载 (http://linux.vbird.org/download/index.php#proftpd)，你可以使用 wget 或者是 ncftp 来到各大 FTP 网站下载喔！

- 设定、编译与安装 proftpd：
  又到了这个时刻了！请注意您的 gcc 以及 make 有没有安装啊！如果没有安装的话，就赶紧先安装吧！我们要来安装 ProFTPD 啰（ 注：我是在 Red Hat 9 上面进行编译测试的！)：

```
1. 将刚刚下载的 proftpd 解压缩：
[root@tet root]# wget  \
>  http://ftp.nsysu.edu.tw/Unix/FTP/proftpd/distrib/source/proftpd-1.2.8.tar.gz
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/proftpd-1.2.8.tar.gz
# ...(略)...会主动产生 proftpd-1.2.8 的目录
```

```
[root@test src]# cd proftpd-1.2.8
# 在这个目录底下的 INSTALL 请详细的阅读，里面有安装的几个重要信息！

2. 进行编译前的参数设定：
[root@test proftpd-1.2.8]# ./configure --prefix=/usr/local/proftpd \
> --enable-shadow --enable-autoshadow  \
> --with-modules=mod_ratio:mod_readme:mod_wrap
# 请注意，那个 prefix 表示我预计要安装 proftpd 的目录;
# 至于 --enable-shadow 与 --enable-autoshadow 则是预计要以
# 系统的 /etc/shadow 做为我的 FTP 登入时的密码验证档案！
# 此外，还加入许多的支持模块，会让我们的 Proftpd 变的更活泼！

[root@test proftpd-1.2.8]# make && make install
# 经过这个步骤之后，你的 proftpd 就会安装在 /usr/local/proftpd 之内，
# 其中，设定档在 /usr/local/proftpd/etc/proftpd.conf ！
# 至于说明档 (man pages)则在 /usr/local/proftpd/man 当中！

3. 设定一些查询的相关功能！
[root@test proftpd-1.2.8]# vi /etc/man.config
# 加入底下这一行，这样才能以 man 来查询指令的用法！
MANPATH /usr/local/proftpd/man
```

- 很简单吧！这样就已经编译并且安装好了 Proftpd 啰！^_^！接下来就要开始来测试看看啰！

- 设定以 xinetd 来启动 proftpd ：
  事实上，目前大部分的 FTP daemon 多是以 super daemon 来启动的！所以这里我们也直接以 xinetd 来设定 proftpd 吧！毕竟多了一层管理，会更安全的啊！ ^_^！设定的方法也真是很简单～只要编辑 xinetd 底下的 proftpd 以及 proftpd.conf 档案里面的一些内容即可！

```
[root@test root]# vi /etc/xinetd.d/proftpd
service ftp
{
  disable        = no
       flags           = REUSE
       socket_type     = stream
       wait            = no
       user            = root
  server         = /usr/local/proftpd/sbin/proftpd
       server_args     = -c /usr/local/proftpd/etc/proftpd.conf
       log_on_success  += DURATION USERID
       log_on_failure  += USERID
```

```
}
# 上面的重点在于两行粗体字的地方！尤其是 server =.. 那一行！
# 由于我们是使用 Tarball 安装的，自然就以我们的档案放置目录为主，
# 如果您使用 RPM 安装，这里就不太相同啰！
# 另外，那个 server_args 后面接的则是设定档的档名咯！

[root@test root]# vi /usr/local/proftpd/etc/proftpd.conf
# 找到底下这个设定值：
ServerType                    standalone
Group                         nogroup
# 因为我们的系统预设并没有 nogroup 这个群组，所以必须要将他改成
# 系统里面有的群组才行呐！而服务器启动的方式有 super daemon 与
# stand alone ，所以：
ServerType                    inetd
Group                         nobody

[root@test root]# /etc/rc.d/init.d/xinetd restart
[root@test root]# netstat -tlnp
tcp        0      0 0.0.0.0:21     0.0.0.0:*     LISTEN   8483/xinetd
# 呵呵！ 21 这个埠口出现了！理论上没有问题，不过还是需要分析一下
# /var/log/messages 这个档案的内容才行喔！
```

- 一般来说，我们还是比较建议使用 xinetd 的方式来启动服务的，尤其是 FTP 之类的服务器架设。在这个案例当中，我们就使用了 xinetd 来启动，不过， proftpd 的设定档 proftpd.conf 里面就必须要规定好 ServerType 这个设定值，有两个选择：

  - standalone：表示独立启动的意思；
  - inetd：表示使用 super daemon ，不论是 inetd 或者是 xinetd 均使用这个设定值。

  所以啰，这里需要修订好才行！然后就可以正确的来启动啰！很简单吧！ ^_^！好了，准备来详细的分析 proftpd 啰！

Server 端设定：

Proftpd 在设定上简单是很简单，不过，如果要说复杂的设定的话，又很多样化喔！那么就来谈一谈吧！

---

proftpd 的结构

这个 proftpd 服务器的架构其实也是很简单的！主要设定档仅有一个而已，我们就来说一说

proftpd 需要注意的档案吧！

- proftpd.conf：这个就是主要的 proftpd 的设定档了！在 RPM 安装的范例中，他所在的目录为 /etc/proftpd.conf ，在鸟哥的这个范例中，则是放置在 /usr/local/proftpd/etc/proftpd.conf 当中喔！我们未来要谈的种种 proftpd 的设定，均是在这个档案里面设定的。他详细的设定方法其实在这个 proftpd.conf 档案里面就已经写得很详细咯!而且 proftpd 也提供了很详细的文件数据了,如果是以 RPM 安装您的 proftpd 的话，那么文件档案放置在 /usr/share/doc/proftpd-"版本"/Configuration.html ，若以 Tarball 安装则在 /usr/local/src/proftpd-"版本"/ Configuration.html。不过，该说明档实在是太复杂了～只要知道其中一些主要设定即可！这也是我们在后续的介绍所想要传达的喔！

- proftpd：这个是主要的 proftpd 的 daemon 执行档！我们得要启动他才行喔！此外，这也是 TCP Wrappers (/etc/hosts.deny(allow))设定里头的服务档案档名。另外，当 proftpd 在启动的时候会去读取设定档，也就是 proftpd.conf 这个档案，不过，我们也可以指定其它的档案来进行 proftpd 的设定喔！果真如此，就必须要这样启动 proftpd 了：

```
[root@test root]# proftpd -c 设定档档名
[root@test root]# proftpd -c /usr/local/proftpd/etc/proftpd.conf
```

-
- ftpcount:目前在主机上面使用 proftpd 的联机数,直接在指令列下达 ftpcount 即可!

- ftpshut:指定再过多久之后 proftpd 服务会终止!有的时候我们会需要维护 FTP 主机对吧！所以需要关机啊！关掉 ftp 服务之前，可以使用这个 ftpshut 指令来进行喔！他的语法如下所示：

```
[root@test root]# ftpshut [-l 分钟] -d [分钟] 时间 "讯息"
参数说明:
-l  : 在 FTP 服务器关闭服务之前的多少分钟，尝试建立新的 FTP 联机者均不被接受
-d  : 在 FTP 服务器关闭服务之前的多少分钟，以建立的 FTP 联机将强制被终止
时间: 在什么时候或多少分钟后， FTP 服务器将关闭 FTP 服务！格式有两种:
      +number : 再经过 number 分钟后 FTP 会关闭
      MMHH    : 在今天的 MM:HH 时间 FTP 会关闭
讯息: 显示给 user 看的信息！
范例:


范例一:
# 再经过 180 分钟后，FTP 会关机，且关机前 20 分钟即不可再接受新联机，
# 而以建立的联机在关机前 10 分钟强制断线，并在 client 端显示:
# FTP will shutdown at time
[root@test root]# ftpshut -l 20 -d 10 +180 "This FTP will shutdown at time"
```

- 事实上， ftpshut 仅会建立一个档案，亦即是 /etc/shutmsg 而已，还记得这个档案吧？！在前一章 Wu FTP 主机设定里面的最简单的 ftpaccess 设定，里头第八项提到的项目，呵呵！没错！就是这个档案咯！如果您想要重新启动 FTP 服务的话，只要将这个档案杀掉，或者是将这个档案里面的相关数字修改一下即可！请翻至前一章节查阅喔！

- ftpwho：可以用来察看目前有多少人使用 proftpd 这个服务喔！简单的语法直接下达 ftpwho 即可，如下所示：

```
[root@test root]# ftpwho
standalone FTP daemon [8451]:
10194 badbird    [ 0m11s]   0m6s (idle)
Service class                      -    1 user
# 如上所示，目前有一个使用者，名为 badbird 的账号，在使用 proftpd 喔！
```

大致上就是这样啦！咦！怎么没有提到 anonymous 登入 FTP 时的根目录呢？呵呵！那个咚咚是在 proftpd.conf 里面设定的啦！等一下再告诉你！

---

proftpd.conf 的设定方式

Proftpd 最重要的设定是在 proftpd.conf 这个档案内了！好了，那么这个档案的内容是如何设定的呢？基本上，这个档案的设定与 Apache 很类似呐！有点像这样：

```
# 关于主机相关的设定
设定项目一  参数内容
设定参数二  参数内容

# 关于某些目录的权限设定
<Directory "完整目录名称">
...
...
...
</Directory>

# 关于 Anonymous 的目录与权限设定
<Anonymous "匿名登入时候的匿名者根目录">
...
...
  <Limit 一些动作>
  ...
  ...
  </Limit>
```

```
</Anonymous>
```

是否与 Apache 主机的设定文件： httpd.conf 语法很类似呢？所以啰，呵呵，设定上也有很相似的参数喔!反正,只要是没有被 <xxx></xxx> 包含在内的设定参数,都是属于主机与 Real User 的设定值，而与匿名者有关的设定则是在 <Anonymous> 与 </Anonymous> 内的设定值！此外，我们还可以透过 <Limit> 这个设定参数来订定某些动作是否可做喔！至于在这个档案内，只要该行是以 # 开头，表示该行是『批注』而已的啦！好了，那么 Limit 有哪些动作呢？！基本上有底下这些：

- CWD ：Change Working Directory，变换目录之意；
- MKD ：MaKe Directory，可建立目录与否；
- RNFR ：ReName FRom，可更改档名与否；
- DELE ：DELEt，可删除档案语法；
- RMD ：ReMove Directory，可移除目录与否；
- RETR ：RETRieve，下载之意！由 Server 传送数据到 Client；
- READ ：可读取与否
- WRITE：可写入与否
- STOR ：STORe，上传之意，由 Client 传送数据到 Server ！
- ALL ：全部的动作！

除此之外，我们还可以指定 .ftpaccess 这个档案的设定呢！这与 proftpd.conf 内的 AllowOverride 参数有关！这个 .ftpaccess 就是允许使用者自行设定 FTP 的风格，当 FTP 的 Client 软件登入某个目录，而该目录内支持 .ftpaccess 时，那么该 FTP Client 软件将接受 .ftpaccess 的使用者自订风格喔!这个 .ftpaccess 与 Apache 的 .htaccess 有类似的用法咯！ ^_^！还有， proftpd.conf 也支持变量，变量的内容如下：

```
%T 目前的时间
%F 所在硬盘剩下的容量
%C 目前所在的目录
%R Client 端的主机名称
%L Server 端的主机名称
%U 使用者账号名称
%M 最大允许联机人数
%N 目前的主机联机人数
%E FTP 主机管理员的 email
%i 本次上传的档案数目
%o 本次下载的档案数量
%t 本次上传+下载的档案数量
```

最简单的 proftpd.conf 设定档

事实上，当我们安装好了 proftpd 之后，就已经提供了一个很简单但是已经够用的 proftpd.conf 的设定内容了！我们就来谈一谈这个简易的设定内容吧！

```
[root@test root]# vi /usr/local/proftpd/etc/proftpd.conf
# 底下是 FTP 主机的环境设定，每个项目的内容为：
# ServerName : 当使用者登入主机的时候，proftpd 会显示在 Client 端
#              的 FTP 软件的一些基本讯息啦！
# ServerType : 启动 proftpd 的方法，有两种方式，分别是 standalone
#              与 inetd ，因为我们是以 super daemon 启动的，所以
#              设定为 inetd 喔！如果您想独立启动(不透过 xinetd )
#              就需要设定为 standalone 了
# DefaultServer：预设的主机啊！这个项目可以设定为 on 或 off ，基本上，
#                除非您有两个 IP 或者是设定了虚拟主机 (virtualhost)，
#                否则这个项目都应该要设定为 on 才行！不然有些 uknown
#                的联机会无法连接到您的 FTP 服务喔！
# Port : 设定主机的 FTP 命令信道端口口！如前面 Wu FTP 所说明的，FTP
#        命令通道通常为 21  ，您也可以更改，不过，这个设定只有当
#        ServerType 为 standalone 时才有效！若为 inetd 则与 xientd 及
#        /etc/services 有关那！请前往参考 wu FTP 的修改 port 设定！
# Umask : 与建立目录及档案的预设属性有关的设定喔！用 022 就够了！
# MaxInstances：同一时间允许的联机数目，这个设定项目与 process (PID) 有关！
#              所以您的 FTP 主机中，proftpd 启用的 process 最多能有 30 个
#              这个与 MaxClients 不一样喔！
# User 与 Group：proftpd 预设的服务启动者！后面接的使用者与群组
#                必须在 /etc/passwd 与 /etc/group 里面存在方可！
ServerName                      "这个是鸟哥的测试用的 Proftp 主机"
ServerType                      inetd
DefaultServer                   on
Port                            21
Umask                           022
MaxInstances                    30
User                            nobody
Group                           nobody

# 底下则是与目录有关的设定！在这个设定中，显示允许读写与覆盖档案！
# AllowOverwrite 就是允许覆写的意思！
<Directory />
  AllowOverwrite                on
</Directory>

# 底下与匿名登入者有关！由 <anonymous ~ftp> 显示：『预设的匿名登入之
# 根目录为 ftp 这个使用者的家目录！』，因为 ~ 代表家目录的意思！
# 而且，匿名登入主机后，该 process 取得的 user:group 权限为 ftp:ftp!
# 至于那个 UserAlias 就是在设定『名字的别名』啊！语法为：
```

```
# UserAlias "登入者的账号" "实际 Linux 主机的账号"
# MaxClients：最多仅允许 10 个 anonymous 登入我们主机的意思！
# DisplayLogin：当使用者登入之后的欢迎画面的档案内容！
# DisplayFirstChdir：转换到某目录时(cd指令)，显示该目录的注意事项档案内容
<Anonymous ~ftp>
  User                          ftp
  Group                         ftp
  UserAlias                     anonymous ftp
  MaxClients                    10
  DisplayLogin                  welcome.msg
  DisplayFirstChdir             .message
  # 底下则是限制 anonymous 『不具有写入的权限！』因为 WRITE 是写入，
  # 加上 DenyAll 则是写入的权限被取消之意！
  <Limit WRITE>
    DenyAll
  </Limit>
</Anonymous>
```

所以，在这个例子当中，我们的 proftpd 具有下列功能：

- 以 super daemon 启动 proftpd ，且启动 proftpd 的权限为 nobody:nobody ，此外，最多仅具有 30 个 process；
- proftpd 使用 port number 为 port 21 喔(其实是需要看 /etc/xinetd.d/proftpd 的设定)！
- 这个 proftpd 同时允许实体用户(real user)与匿名用户(anonymous)登入！
- real user 可以跳离自己的家目录(没有 chroot)，且具有写入的权限，此外，建立目录与档案预设权限为 dir:755, file:644;
- anonymous 用户最多仅能同时 10 人上线，且具有的权限为 ftp:ftp，并且『anonymous 没有写入的权限！』

事实上，这样的设定已经能够符合一般主机的设定啰！如果您还不满意，可以额外的再加入其它的设定喔！底下就来谈一谈！

针对实体用户的设定

事实上，在上面的 最简单 proftpd.conf 设定 当中就已经针对了实体用户(Real User)进行了若干的设定了！不过，如果您还想额外的加入某些设定，那么就继续来修改吧！假设我们需要这样的设定项目：

- 使用主机本地端时间，而不要使用 GMT 时间；

- 主机最多允许 50 条联机，且最多允许 100 个使用者上线，当超过 100 个使用者还有其它 FTP 要求时，就显示：『很抱歉，上线人数额满了！』；
- 同一个 IP (或主机) 来源最多仅能具有 5 个 FTP 服务；
- 允许续传的动作；
- 被动式资料流(passive mode)的埠口为 65400 到 65420 这 21 个埠口！
- 主机不允许 Root 登入；
- 想建立一个名为 badbird 的群组，在该群组内的所有使用者都无法离开自己的家目录(chroot)；
- 在 badbird 这个群组当中的 nogoodbird 这个使用者，该使用者能够使用 ftp 但是无法使用 ssh 连到主机；
- 在这个公开的目录 /home/ftp/pub 中，所有人均不可写入，只有读取的权限；

```
[root@test root]# vi /usr/local/proftpd/etc/proftpd.conf
# 底下是 FTP 主机的环境设定：
ServerName                      "这个是鸟哥的测试用的 Proftp 主机"
ServerType                      inetd
DefaultServer                   on
Port                            21
User                            nobody
Group                           nobody
TimesGMT                        off  # 所谓的 GMT 时间就是格林威治时间，
                                     # 详细的时区观念请参考后续的NTP服务器
                                     # 因为要使用本地时间，所以设为 off ！
MaxInstances                    50   # 最多仅有 50 条 prftpd 的 PID
MaxClients  100  "很抱歉上线人数额满了" # 最多允许 100 个使用者在在线
MaxClientsPerHost               5    # 同一个主机最多可以同时 5 个 FTP 联机
AllowStoreRestart               on   # 允许使用者上传续传！预设是 off
PassivePorts   65400  65420
# 后面接的是埠口，最小到最大的埠口共 21 个！

# 其它与实体用户较相关的设定值！
Umask                           022
RootLogin                       off  # 不许 root 登入！预设就是 off
RequireValidShell               off
# 这个设定可以让使用者不需要具有『能够执行的 shell 』！例如让
# nogoodbird 这个具有 /bin/false 的使用者，依然可以使用 ftp 喔！
DefaultRoot                     ~ badbird
# 注意啊！那个 ~ 代表家目录的意思喔！特别特别留意！ DefaultRoot
# 后面接的是『群组』喔！所以在这里 badbird 为群组，而不是使用者喔！
# 这里特别容易搞混乱，请再特别的留意一下阿！只要不属于  badbird
# 这个群组的 User 就可以离开自己的家目录了！(没有被 chroot ) ^_^
<Directory />
  AllowOverwrite                on
```

```
</Directory>
<Directory /home/ftp/pub>
  <Limit WRITE>
  Denyall
  </Limit>
</Directory>
# 上面的设定中，在根目录内的所有目录均具有可擦写的权力，但是在
# /home/ftp/pub 这个目录中，不论 Linux 属性为何，使用者均无法写入!
# 但是可以浏览以及下载喔! 在我们这个设定当中， badbird 这个群组无法离开
# 自己的家目录，至于其它可以离开自己家目录的使用者，来到这个
# /home/ftp/pub 当中，也不具有写入的权限喔!

[root@test root]# useradd -g badbird -m -s /bin/false nogoodbird
# 建立这个 nogoodbird 由于不具有 shell 所以不能 SSH 但可以 ftp 喔!

[root@test root]# /etc/rc.d/init.d/xinetd restart
```

事实上，对于实体用户实在不需要限制的太多！要不然就不要开放，要不然就直接改成 sftp 说！此外，在上面这个设定当中，我们暂时拿掉了 anonymous 的登入，所以使用 anonymous 将无法登入喔！

针对匿名者的设定

谈完了实体用户之后，我们来谈一谈，那么 anonymous 的相关登入权限要怎么设定呢？！我们的要求假设如下：

- 主机环境与实体用户的需求与上面相同；
- anonymous 的根目录为 /var/ftp 这个目录；
- anonymous 登入后取得的 PID 在 Linux 的权限为 ftp:ftp 这个人物；
- 当 anonymous 登入 FTP 之后，在 Client 端的 FTP 软件显示一些欢迎讯息！；
- 最多允许 30 个 anonymous 的登入；
- 限制上传/下载速度为 100Kbytes/s 与 50 Kbytes/s；
- 在 /var/ftp/ 里面，除了 /var/ftp/upload 之外，其它的目录均不可写入；
- 在 /var/ftp/upload 这个目录中，仅可以写入，不能下载，并且在使用者进入这个目录后，显示出一些相关的信息；
- 使用者账号为 nogoodbird 因为不乖，所以将他们的 FTP 使用权限降级而设定为 anonymous 而已！

如何设定呢？！我们沿用上面的设定项目，再额外的新增底下的项目即可！

```
1. 建立基本的设定档案:
[root@test root]# vi /usr/local/proftpd/etc/proftpd.conf
# 关于主机与实体用户的设定如同前一小节所示，所以我这里就略过了!
...(沿用上一小节的设定，这里略过).....

# 底下则是 anonymous 的设定喔!
<Anonymous /var/ftp>
  # 底下为建立 Anonymous 在 Linux 系统下的 PID 权限拥有者!
  # 此外，使用 UserAlias 将 nogoodbird 降级为 anonymous 的账号!
  User                ftp
  Group               ftp
  UserAlias           anonymous ftp
  UserAlias           nogoodbird ftp
  # 建立显示的讯息给 anonymous 观察用的!
  DisplayLogin        welcome.msg
  DisplayFirstChdir   .message
  MaxClients          30  "匿名登入者联机数已经饱和了! "
  # 这个就重要啦! 用来限制传输速率的呐! 基本语法为:
  # TransferRate (STOR|RETR) 速度(Kbytes/s) user 使用者
  # STOR 为上传而 RETR 为下载的意思! 速度为 Kbytes/second 喔!
  TransferRate  STOR    100   user anonymous,ftp # 单位为 KBytes/second
  TransferRate  RETR    50    user anonymous,ftp
  <Limit WRITE>
    Denyall
  </Limit>
  # 底下这个则仅与 upload 这个目录以及其下的子目录有关而已!
  <Directory /var/ftp/upload/*>
    <Limit READ>
      Denyall
    </Limit>
    <Limit WRITE>
      Allowall
    </Limit>
  </Directory>
</Anonymous>

2. 建立欢迎画面:
# 特别留意，因为我的 anonymous 根目录在 /var/ftp ，因此，我的
# welcome.msg 就必须放置在 /var/ftp/welcome.msg 了!
[root@test root]# vi /var/ftp/welcome.msg
欢迎光临! 这个是鸟哥的测试 FTP 站台喔!
我的主机: %L
目前时间: %T
最大联机: %M
```

```
目前联机： %N
您的主机： %R
您的账号： %U
目前目录： %C

3. 建立特殊注意事项：
# 刚刚提到，需要在 /var/ftp/upload 里面建立一个特殊讯息！
[root@test root]# vi /var/ftp/upload/.message
这个目录仅能上传不能下载；
您的身份为 anonymous 喔！

4. 建立 upload 的权限：
[root@test root]# chown ftp:ftp /var/ftp/upload
[root@test root]# chmod 755 /var/ftp/upload

5. 重新启动！
[root@test root]# /etc/rc.d/init.d/xinetd restart
```

呵呵！这样就将您的 Anonymous 设定好了！等你一进站，哇！怎么这么棒啊！已经将您的信息都给他设定好了，欢迎画面可真的是不错啊！ ^_^

建立特殊交流账号 (建立一个 ftpguest 群组！将所有的 guset 设定在这个群组内！

我想，很多朋友都有使用 FTP 网站与其它网站交流的经验了！您可以给予某些站长一些上传与下载的权限，并且这些权限是可以保留或者是累积的，真的是很棒啊！在 Windows 系统上面有 Server-U 这个好用的家伙，那么我们的 Linux 上头的 FTP 可以达到这样的功能吗？！呵呵！proftpd 就可以！而且设定还真的是很简单喔！假设我们要达成这样的功能好了：

- o 主机环境、实体用户、anonymous 的环境都与前两节的内容相同；
- o 建立一个群组名为 ftpguest ，如果使用者属于该群组，则该使用者登入主机之后他的根目录会在 /var/ftp2 这个目录下；
- o 有三个使用者，名为 ftpuser1, ftpuser2, ftpuser3 ，都属于 ftpguest 群组，他们没有家目录，不能使用 ssh，但是他们在 /var/ftp2/upload 有写入的权限，但不可读取数据；
- o 在 /var/ftp2 内的所有相关下载中，最高流量为 50 Kbytes/second；
- o ftpuser1 的上传/下载比例为 1:2 ，且具有 100 MB 的预设下载量； ftpuser2 与 ftpuser3 的上传/下载比例则为 1:1，仅具有 30MB 的预设下载量；
- o 当使用者进入 /var/ftp2 时，会显示该使用者的上传/下载比例，以及剩下的下载容量，还有其它的相关讯息；
- o 与使用者有关的上传/下载比例以及剩下的可下载容量，都记录在 /var/ftp2/work/ratio.dat, /var/ftp2/work/ratio.tmp 当中，所以使用者在这个目录都无法读、写！

在这个案例当中，最重要的就是那个『纪录使用者上传/下载的 ratio 以及可用空间的记录文件』了，在我的案例当中，使用的就是 /var/ftp2/work/ratio.dat 这个档案，请注意，这个档案必须要能被 ftpuser1, ftpuser2, ftpuser3 所读取与写入才行！相当的重要喔！所以，我应该要这样设计我的设定档：

```
1. 建立所需要的群组与使用者：
#  我要建立一个群组为 ftpguest ，此外，所有相关的使用者都是这个群组！
[root@test root]# groupadd ftpguest
[root@test root]# useradd -M -g ftpguest -s /bin/false ftpuser1
[root@test root]# useradd -M -g ftpguest -s /bin/false ftpuser2
[root@test root]# useradd -M -g ftpguest -s /bin/false ftpuser3
[root@test root]# passwd ftpuser1
# 请依序建立 ftpuser1 ftpuser2 ftpuser3 的密码！

2. 建立所需要的 FTP 相关路径：
#  我要的路径在 /var/ftp2 当中，而且 ftpguest 必须要能够写入！
[root@test root]# mkdir -p /var/ftp2
[root@test root]# mkdir -p /var/ftp2/upload
[root@test root]# mkdir -p /var/ftp2/work
[root@test root]# chmod -R 775 /var/ftp2
[root@test root]# touch /var/ftp2/work/ratio.dat #底下两个档案用在 ratio
[root@test root]# touch /var/ftp2/work/ratio.tmp
[root@test root]# chown -R ftpuser1:ftpguest /var/ftp2
[root@test root]# chmod 666 /var/ftp2/work/*

3. 建立基本的设定档案：
[root@test root]# vi /usr/local/proftpd/etc/proftpd.conf
# 关于主机,实体用户,anonymous 的设定如同前两小节所示，所以我这里就略过了！
...(沿用上两小节的设定，这里略过).....

# 底下则是 /var/ftp2 的设定喔！就是与 ftpguest 有关的设定喔！
DefaultRoot            /var/ftp2 ftpguest
DisplayLogin           welcome.msg
# 开始设定上传/下载比例
Ratios                 on
SaveRatios             on
RatioFile              /work/ratio.dat
RatioTempFile          /work/ratio.tmp
# 上面这两个档案需要比较注意！他的路径与 DefaultRoot 有关系！
# 因为我们的 DefaultRoot 在 /var/ftp2 ，因此，这个档案在
# 『根目录为 /var/ftp2 时，路径为 /work/』也就是说， /work/ratio.dat
# 其实就是 /var/ftp2/work/ratio.dat (因为 / 是 /var/ftp2 喔)
# 这个地方是最容易搞错的！请再次的看清楚喔！ ^_^
```

```
# 至于底下的设定就是要让 /var/ftp2/work 这个目录下的档案都无法被使用!
<Directory /var/ftp2/work>
  <Limit All>
    Denyall
  </Limit>
</Directory>

# 这里就是在设定使用者的上传/下载比例啦! 语法为:
# UserRatio  "使用者账号"  fileratio filequota byteratio bytequota
#   使用者账号: 就是登入 proftpd 的账号啊!
#   fileratio : 这个是以档案为基准的『比例』, 通常不限制, 故为 0
#   filequota : 预设能够下载多少档案, 不限制时为 0
#   byteratio : 就是上传/下载的比例, 这个数字代表『 1:下载 』之意!
#   bytequota : 预设能够下载多少 KBytes 的档案! 注意单位喔!
UserRatio               ftpuser1  0  0 2 100000 # 上/下比例为 1:2
UserRatio               ftpuser2  0  0 1  30000
UserRatio               ftpuser3  0  0 1  30000
# UserRatio             ftpuser3  0  0 -2  30000
# 上面这行有意思! 当下载比例为负值时, 表示上/下 比例为 2:1 的意思!
<Directory /var/ftp2>
  Umask                 002
  # 这里就是在进行『下载速度的限制』啰!
  TransferRate  RETR    50  group  ftpguest
  <Limit WRITE>
    Denyall
  </Limit>
</Directory>

<Directory /var/ftp2/upload/*>
  <Limit READ>
    Denyall
  </Limit>
  <Limit WRITE>
    Allowall
  </Limit>
</Directory>

4. 建立欢迎画面:
# 特别留意, 因为我的 ftpguest 群组的根目录在 /var/ftp2 , 因此, 我的
# welcome.msg 就必须放置在 /var/ftp2/welcome.msg 了!
[root@test root]# vi /var/ftp2/welcome.msg
欢迎光临! 这个是鸟哥的测试 FTP 站台喔!
我的主机:  %L
```

```
目前时间： %T
最大联机： %M
目前联机： %N
您的主机： %R
您的账号： %U
目前目录： %C

5. 重新启动!
[root@test root]# /etc/rc.d/init.d/xinetd restart
```

这样就设定妥当，并且也可以正确的启用啰！好了！那么我们就赶紧来测试看看能不能记录每个使用者的上传/下载比例呢？如下所示：

```
[root@test ftp2]# ftp localhost
Connected to localhost (127.0.0.1).
220 ProFTPD 1.2.8 Server (这个是鸟哥的测试用的 Proftp 主机) [test.localhost]
Name (localhost:root): ftpuser1
331 Password required for ftpuser1.
Password:  <== 这里输入 ftpuser1 的密码
230-欢迎光临！这个是鸟哥的测试 FTP 站台喔!
 我的主机:  test.localhost
 目前时间:  Fri Sep  5 01:08:10 2003
 最大联机:  100
 目前联机:  1
 您的主机:  localhost.localdomain
 您的账号:  ftpuser1
 目前目录:  /
230-User ftpuser1 logged in.
230 Down: 0 Files (0mb)  Up: 0 Files (0mb)   1:2B CR: 97
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bye
```

看到上面的粗体字了吧？那表示：

- Down：下载方面，目前下载过 0 个档案，共下载 0 MBytes;
- Up ：上传方面，目前上传过 0 个档案，共上传 0 MBytes ;
- 1:2B：上传/下载 (为 Bytes 限制)的比例为 1:2 咯！也就是上传 1MB 可以下载 2MB 咯！
- CR ：剩余的可以下载的 MBytes 数！注意单位为 MBytes 喔！

未来您只要有上传或者是下载，那么上面那行粗体自就会随之而变喔！事实上，这些信息是记录

在 proftpd.conf 里面的参数『RatioFile』指定档案当中！您可以检查一下该档案：

```
[root@test ftp2]# vi /var/ftp2/work/ratio.dat
ftpuser1|0|0|0|0
# 这个档案的格式为：(注：以符号『|』隔开各个字段)
# 账号|上传档案数|上传档案总量(KBytes)|下载档案数|下载档案总量(KBytes)
```

该档案的格式如同上面的说明，以符号『 | 』来隔开成为五个字段，每次使用者登入 proftpd 主机之后， Proftpd 会先去开启这个档案，将数据读出来，然后再与 UserRatio 的设定直比较，就可以持续的纪录每个使用者的剩余可下载容量啰！那么未来如果想要新增其它使用者在这个设定群组当中，只要进行(1)新增使用者，注意这个使用者的群组需要是 ftpguest ，并且(2)再到 proftpd.conf 里面设定好 UserRatio 的设定值，(3)最后重新启动 xinetd ，就 OK 了！其它的步骤不需要进行！设定是否方便很多呢！？^_^

小结语

就鸟哥的感觉来看， Proftpd 真的是挺不错用的，编译上并不难，设定上也挺简单的，此外，还能够提供类似 Windows 里面那个 Server-U 的档案上传/下载比例，真的是很不错，尤其要制作 upload 的目录真的是太简易了！加上他的安全性又比 Wu FTP 来的好，实在是一个不错的 FTP 服务器软件啊！如果您对于 Windows 的系统不太满意，又喜欢 Server-U 的设定内容，那么建议您可以转换 Linux 的 proftpd 来尝试看看，真的是挺好玩的喔！ ^_^

Client 端的设定：
Client 端并没有什么好设定的地方，主要就是 ftp 的使用了，请参考 wu FTP 主机设定一节！

参考资源

- ProFTP 官方网站： http://www.proftpd.org
- proftpd.conf 的官方说明文件：
  http://www.proftpd.org/docs/directives/linked/by-name.html
- proFTP 官方网站的一些范例：http://www.proftpd.org/docs/example-conf.html
- 您的主机内的说明文件：/usr/share/doc/proftpd-"version"/Configuration.html

---

课后练习

- 如何建立一个使用者，他可以使用 FTP 的功能，但是无法以 telnet 或 ssh 登入系统？！请使用 proftpd 的设定项目来设定！
- 我明明在台湾，我的主机时区 (/etc/sysconfig/clock) 明明在台湾，为何登入 proftpd 之后，显示的系统时间就是慢了 8 小时？请问为什么？如何解决？
- 如果发生了无法登入，或者是与 proftpd 的 FTP 功能相关的错误时，要如何 debug 呢？
  前往参考用解答

简易 vsftpd 服务器设定

最近更新日期：2003/09/04

既然说 FTP 真的很危险，那么不要开启也罢！不过，偏偏很多时候我们又得使用到 FTP 的功能。这个时候选择一个比较安全的 FTP 服务器软件就很重要啦！除了 proftpd 可以取代 Wu FTP 之外，另一个蛮安全的 vsftpd 也可以用来取代喔！这个章节我们就来谈一谈 vsftpd 的设定吧！

前言

除了 proftpd 之外，事实上，这个 vsftpd 也是一个很好的 FTP 服务器软件喔！为什么这么说呢？因为 vsftpd 全名是『very secure FTP daemon』的意思，所以他的发展本来就是以安全性为考虑来发展这个套件的。vsftpd 在安全性的考虑上面，主要针对了『程序的权限，privilege』概念来设计的，因为我们的任何服务在 Linux 上面运作时都会取得一个 PID，而这个 PID 是有拥有者的身份的，也就是说，这个服务的 PID 在我们的 Linux 上面是具有某些『权限』的。万一这个服务的 PID 所属拥有者的身份等级太高，例如 root 的权限，那么如果不幸该 PID 有些设计上的漏洞，使得该 PID 被入侵的话，入侵者将具有该 PID 的权限，也就是 root 的身份喔！所以，近来发展的套件都会尽量的将服务取得的 PID 权限降低，使得该服务即使不小心被入侵了，入侵者也无法得到有效的系统管理权限，这样会让我们的系统较为安全的啦。

除了上面这个权限的设计之外， vsftpd 也利用 chroot 这个软件的辅助，来让登入者仅能于一些较不重要的目录当中活动，而无法使用 Linux 系统全部的功能。所谓的 chroot 这个函数，最主要的功能就是『改变根目录的所在 (change root directory)』了！举例来说，如果您想要让使用者登入 A 服务后，且执行任何指令都是在 /tmp/pub 目录下，并限制使用者使用 A 服务时都只能在 /tmp/pub 目录下，那么使用『 chroot /tmp/pub command 』就能够让 /tmp/pub 变成 A 服务的根目录『/』了！如此一来，使用者就无法离开 /tmp/pub ，那么万一我们 A 服务的 PID 还是被入侵时，没有关系，入侵者还是仅能在 /tmp/pub 里面跑来跑去而已，而无法使用 Linux 的完整功能。这个时候，自然我们的系统也就会比较安全啦！

vsftpd 是基于上面的说明来设计的一个较为安全的 FTP 服务器软件，他具有底下的特点喔：

- vsftpd 是以一般身份启动服务，所以对于 Linux 系统的使用权限较低，对于 Linux 系统的危害就相对的减低了。此外， vsftpd 亦利用 chroot() 这个函式进行改换根目录的动作，使得系统工具不会被 vsftpd 这支服务所误用；
- 任何需要具有较高执行权限的 vsftpd 指令均以一支特殊的上层程序 (parent process) 所控制 ，该上层程序享有的较高执行权限功能已经被限制的相当的低，并以不影响 Linux 本身的系统为准；
- 所有来自 clients 端，想要使用这支上层程序所提供的较高执行权限之 vsftpd 指令的需求，均被视为『不可信任的要求』来处理，必需要经过相当程度的身份确认后，方可利用该上层程序的功能。例如 chown(), Login 的要求等等动作；
- 此外，上面提到的上层程序中，依然使用 chroot() 的功能来限制使用者的执行权限。

由于具有这样的特点，所以 vsftpd 会变的比较安全一些咯！

另外，要架设 vsftpd 之前，还是请您得先要针对 FTP 的主动联机、被动联机以及 port 21, 20 这两个指令信道与数据信道的基础有一定程度的认识喔，会比较容易进入状况，所以，还是回到前面的 Wu FTP 那一张节，将前言的部分看完才好呐！我这里假设您已经具有 FTP 的相关知识了，所以底下就直接来进行 vsftpd 的安装与设定吧！

套件安装：

以 RPM 安装

在目前新版的 Red Hat 9 主要的 FTP 服务器软件就是 vsftpd 这个玩意儿！所以您可以拿出光盘里面的 vsftpd 来直接以 RPM 安装即可！如果您的 Linux distribution 没有提供 vsftpd 的话，没有关系，我们也可以使用底下的 Tarball 的方式来安装呐！

以 Tarball 安装

要以 Tarball 安装，当然得先下载 Tarball 的档案了！vsftpd 的官方网站下载点为：

ftp://vsftpd.beasts.org/users/cevans/

您可以自行找寻自己喜欢的版本来安装。我这里以 1.2.0 这一版来安装 vsftpd 在我的 Mandrake 9.0 上面喔！(注：如果是 Red Hat 的系统，原本就有 vsftpd 了，所以使用 RPM 安装比较好！至于其它没有提供 vsftpd RPM 档案的 distribution 就可以使用 Tarball 咯！)

```
1. 下载与解压缩：
[root@test root]# wget \
> ftp://vsftpd.beasts.org/users/cevans/vsftpd-1.2.0.tar.gz
[root@test root]# cd /usr/local/src
[root@test root]# tar -zxvf /root/vsftpd-1.2.0.tar.gz
```

```
[root@test root]# cd vsftpd-1.2.0/
# 在这个目录下有个 INSTALL 与 README 请务必察看喔!

2. 开始编译与安装
# vsftpd 预设安装的路径为:
# 所有可执行档放置在 /usr/local/sbin 里面;
# man page 放置在 /usr/local/man/man5 与 /usr/local/man/man8
# 若 super daemon 为 xinetd 时，会复制一份启动档案到 /etc/xinetd.d 去!
[root@test vsftpd-1.2.0]# make
# 编译的过程可能有 warning 的讯息，只要不是 Error 就可以不理他!
[root@test vsftpd-1.2.0]# make install
[root@test vsftpd-1.2.0]# cp vsftpd.conf /etc
# 将 PAM 身份认证模块给他放进去系统里面!
[root@test vsftpd-1.2.0]# cp RedHat/vsftpd.pam /etc/pam.d/vsftpd
# 建立 ftp 这个使用者以及他的家目录:
# 若本来就存在 ftp 这个使用者，那就不需要进行新增!
[root@test vsftpd-1.2.0]# useradd -M ftp -d /var/ftp
[root@test vsftpd-1.2.0]# mkdir -p /var/ftp
[root@test vsftpd-1.2.0]# chown root:root /var/ftp
[root@test vsftpd-1.2.0]# chmod 755 /var/ftp
# 建立 vsftpd 需要的特殊目录
[root@test vsftpd-1.2.0]# mkdir -p /usr/share/empty

3. 如果需要移除时:
# 如果想要移除 vsftp 时，可以这样做
[root@test vsftpd-1.2.0]# rm /usr/local/sbin/vsftpd
[root@test vsftpd-1.2.0]# rm /usr/local/man/man5/vsftpd.conf.5
[root@test vsftpd-1.2.0]# rm /usr/local/man/man8/vsftpd.8
[root@test vsftpd-1.2.0]# rm /etc/xinetd.d/vsftpd
[root@test vsftpd-1.2.0]# rm /etc/vsftpd.conf
# 因为刚刚安装只有安装这几个档案而已说! 所以啦， vsftpd 真的是挺安全的说!

4. 测试:
# 先确认一下 xinetd.d 有没有问题再说:
[root@test root]# vi /etc/xinetd.d/vsftpd
service ftp
{
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/local/sbin/vsftpd
        log_on_success          += DURATION USERID
        log_on_failure          += USERID
        nice                    = 10
```

```
    disable                 = no
}
[root@test root]# /etc/rc.d/init.d/xinetd restart
[root@test root]# ftp localhost
ftp localhost
Connected to localhost.
220 (vsFTPd 1.2.0)
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (localhost:root): anonymous
# 这样就表示 vsftpd 已经可以正确的启动了，不过因为我们还没有设定好
# /etc/vsftpd.conf ，所以会有无法登入的问题！没关系，
# 等一下设定好就 OK 了！
```

安装的过程真的是很简单，不过， vsftpd.conf 这个档案放置的地点在 RPM 与 Tarball 则可能有点不一样，需要给他特别留意呢！例如 Red Hat 9 预设放置在 /etc/vsftpd/vsftpd.conf ，而 Tarball 则预设放置在 /etc/vsftpd.conf 里面说！

Server 端的设定

其实在 Server 端的设定蛮容易的，因为整个 vsftpd 的设定档几乎可以说只有一个，那就是 vsftpd.conf 这个档案了。底下我们就来谈一谈整个 vsftpd 的套件结构与如何设定编辑 vsftpd.conf 这个设定档吧！ ^_^

---

vsftpd 的套件结构

vsftpd 的套件结构很简单，设定文件与执行档实在是不多，无论如何，我们还是得要了解一下：

- /etc/vsftpd.conf 或 /etc/vsftpd/vsftpd.conf：这个就是 vsftpd 的主要设定档了！也是等一下我们要设定的主要项目说。在这个设定文件里面，所有的设定项目都是以『参数=设定值』来设定的，注意一下，等号两边没有空白喔！至于 vsftpd.conf 的详细说明，其实在 vsftpd.conf 里面就已经相当的清晰了，如果还想要有其它的支持，可以使用『man 5 vsftpd.conf』来查阅喔！

- /etc/pam.d/vsftpd 与 /etc/ftpusers 或 /etc/vsftpd.ftpusers：这个与 Wu FTP 是相同的作用啦！利用 pam 模块来进行身份确认的动作说！那么怎么知道使用 /etc/vsftpd.ftpusers 或 /etc/ftpusers 呢？看 /etc/pam.d/vsftpd 的内容即可！

```
[root@test root]# vi /etc/pam.d/vsftpd
#%PAM-1.0
auth       required     pam_listfile.so item=user sense=deny file=/etc/vsftpd.ftpusers
```

```
onerr=succeed
auth       required     pam_stack.so service=system-auth
auth       required     pam_shells.so
account    required     pam_stack.so service=system-auth
session    required     pam_stack.so service=system-auth
```

- 上面的斜体字是同一行,注意到斜体字那一行,可以发现 file="檔名" 那个档名就是『限制使用者无法使用 vsftpd 』的主要设定档案啰!
- /etc/vsftpd.chroot_list:这个档案不见得会使用到,且与实体用户有关!当我们在 vsftpd.conf 里面设定好了实体用户的使用者没有被 chroot 到自己的家目录下(也就是使用者登入后不只能到自己的家目录,还可以跳到其它目录),不过,某些使用者您想让他无法离开家目录时,预设在 /etc/vsftpd.chroot_list 这个档案里面,就可以将该使用者限制在自己的家目录内了!一行一个账号。
- /etc/vsftp.banned_emails:这个档案与匿名登入有关,不过也不见得会用到!当您允许匿名者 (anonymous) 登入您的 FTP 主机,不过却不允许某些 email address 登入,那么就可以将该 email address 写入到这个档案里面去喔!
- /usr/local/sbin/vsftpd 或 /usr/sbin/vsftpd:这就是 vsftpd 的主要执行档咯!不要怀疑, vsftpd 只有这一个执行档而已啊!
- /var/ftp:这个是 vsftpd 的预设匿名者登入的根目录喔!

大致上就只有这几个档案需要注意而已呢!

---

vsftpd.conf 设定值说明

vsftpd.conf 是 vsftpd 的主要设定档案,在这里我们约略来说明一下常见的 vsftpd.conf 里面的各个设定参数吧!

```
关于主机的设定值
connect_from_port_20=YES (NO)
  还记得 wu ftp 那篇文章提到的,关于主动联机的 ftp-data 吗?
  这个设定项目在启动主动联机的 port 20 咯!
listen_port=21
  使用的 vsftpd 命令通道的 port number 设定,如果您想要使用非
  正规的 ftp port,在这个设定项目修改吧!
dirmessage_enable=YES (NO)
  当使用者进入某个目录时,会显示该目录需要注意的内容,显示的
  档案预设是 .message ,当然,可以使用底下的设定项目来修订!
```

```
message_file=.message
  当 dirmessage_enable=YES 时，可以设定这个项目来让 vsftpd
  寻找该档案来显示讯息！您也可以设定其它档名喔！
listen=YES (NO)
  若设定为 YES 表示 vsftpd 是以 standalone 的方式来启动的！
pasv_enable=YES (NO)
  启动被动式联机(passive mode)，一定要设定为 YES 的啦！
use_localtime=YES (NO)
  是否使用主机的时间？！预设使用 GMT 时间(格林威治)，会比台湾
  时间晚 8 小时，一般来说，建议设定为 YES 吧！
write_enable=YES (NO)
  是否允许使用者具有写入的权限？！这包括删除与修改等功能喔！
connect_timeout=60
  单位是秒，如果 client 尝试连接我们的 vsftpd 命令通道超过 60 秒，
  则不等待，强制断线咯。
accept_timeout=60
  当使用者以被动式 PASV 来进行数据传输时，如果主机启用 passive port
  并等待 client 超过 60 秒，那么就给他强制断线！您可以修改 60 这个数值。
data_connection_timeout=300
  如果 client 与 Server 间的数据传送在 300 秒内都无法传送成功，
  那 Client 的联机就会被我们的 vsftpd 强制剔除！
idle_session_timeout=300
  如果使用者在 300 秒内都没有命令动作，强制离线！
max_clients=0
  如果 vsftpd 是以 stand alone 方式启动的，那么这个设定项目可以设定
  同一时间，最多有多少 client 可以同时连上 vsftpd 哩！？
max_per_ip=0
  与上面 max_clients 类似，这里是同一个 IP 同一时间可允许多少联机？
pasv_max_port=0
pasv_min_port=0
  上面两个是与 passive mode 使用的 port number 有关，如果您想要使用
  65400 到 65410 这 11 个 port 来进行被动式资料的连接，可以这样设定
  pasv_max_port=65410 以及 pasv_min_port=65400
ftpd_banner=一些文字说明
  当使用者无法顺利连上我们的主机，例如联机数量已经超过 max_clients
  的设定了，那么 client 的画面就会显示『一些文字说明』的字样，您可以修改

关于实体用户登入者的设定值
guest_enable=YES (NO)
  若这个值设定为 YES 时，那么任何非 anonymous 登入的账号，均会被
  假设成为 guest (访客) 喔！
local_enable=YES (NO)
  这个设定值必须要为 YES 时，在 /etc/passwd 内的账号才能以
  实体用户的方式登入我们的 vsftpd 主机喔！
```

```
local_max_rate=0
  实体用户的传输速度限制，单位为 bytes/second， 0 为不限制。
chroot_local_user=YES (NO)
  将使用者限制在自己的家目录之内(chroot)！这个设定在 vsftpd
  当中预设是 NO，因为有底下两个设定项目的辅助喔！
  所以不需要启动他！
chroot_list_enable=YES (NO)
  是否启用将某些实体用户限制在他们的家目录内？！预设是 NO ，
  不过，如果您想要让某些使用者无法离开他们的家目录时，
  可以考虑将这个设定为 YES ，并且规划下个设定值
chroot_list_file=/etc/vsftpd.chroot_list
  如果 chroot_list_enable=YES 那么就可以设定这个项目了！他里面可以规定
  那一个实体用户会被限制在自己的家目录内而无法离开！(chroot)
  一行一个账号即可！
userlist_deny=YES (NO)
  若此设定值为 YES 时，则当使用者账号被列入到某个档案时，在该档案内
  的使用者将无法登入 vsftpd 服务器！该档案文件名与下列设定项目有关。
userlist_file=/etc/vsftpd.user_list
  若上面 userlist_deny=YES 时，则这个档案就有用处了！在这个档案内的
  账号都无法使用 vsftpd 喔！

关于匿名者登入的设定值
anonymous_enable=YES (NO)
  设定为允许 anonymous 登入我们的 vsftpd 主机！预设是 YES ，底下的所有
  相关设定都需要将这个设定为 anonymous_enable=YES 之后才会生效！
anon_world_readable_only=YES (NO)
  仅允许 anonymous 具有下载可读档案的权限，预设是 YES。
anon_other_write_enable=YES (NO)
  是否允许 anonymous 具有写入的权限？预设是 NO！如果要设定为 YES，
  那么开放给 anonymous 写入的目录亦需要调整权限，让 vsftpd 的 PID
  拥有者可以写入才行！
anon_mkdir_write_enable=YES (NO)
  是否让 anonymous 具有建立目录的权限？默认值是 NO！如果要设定为 YES，
  那么 anony_other_write_enable 必须设定为 YES ！
anon_upload_enable=YES (NO)
  是否让 anonymous 具有上传数据的功能，预设是 NO，如果要设定为 YES ，
  则 anon_other_write_enable=YES 必须设定。
deny_email_enable=YES (NO)
  将某些特殊的 email address 抵挡住，不让那些 anonymous 登入！
  如果以 anonymous 登入主机时，不是会要求输入密码吗？密码不是要您
  输入您的 email address 吗？如果你很讨厌某些 email address ，
  就可以使用这个设定来将他取消登入的权限！需与下个设定项目配合：
banned_email_file=/etc/vsftpd.banned_emails
  如果 deny_email_enable=YES 时，可以利用这个设定项目来规定那个
```

```
  email address 不可登入我们的 vsftpd 喔！在上面设定的档案内，
  一行输入一个 email address 即可！
no_anon_password=YES (NO)
  当设定为 YES 时，表示 anonymous 将会略过密码检验步骤，
  而直接进入 vsftpd 服务器内喔！所以一般预设都是 NO 的！
anon_max_rate=0
  这个设定值后面接的数值单位为 bytes/秒 ，限制 anonymous 的传输速度，
  如果是 0 则不限制(由最大频宽所限制)，如果您想让 anonymous 仅有
  30 KB/s 的速度，可以设定『anon_max_rate=30000』
anon_umask=077
  限制 anonymous 的权限！如果是 077 则 anonymous 传送过来的档案
  权限会是 -rw------- 喔！

关于系统安全的设定值：
ascii_download_enable=YES (NO)
  如果设定为 YES ，那么 client 就可以使用 ASCII 格式下载档案。
  一般来说，由于启动了这个设定项目可能会导致 DoS 的攻击，因此预设是 NO。
ascii_upload_enable=YES (NO)
  与上一个设定类似的，只是这个设定针对上传而言！预设是 NO。
async_abor_enable=YES (NO)
  如果您的 FTP client 会下达 "async ABOR" 这个指令时，这个设定才需要启用
  一般来说，由于这个设定并不安全，所以通常都是将他取消的！
check_shell=YES (NO)
  如果您想让拥有任何奇怪的 shell 的使用者(在 /etc/passwd 的 shell 字段)
  可以使用 vsftpd 的话，这个设定可以设定为 NO 喔！
one_process_model=YES (NO)
  这个设定项目比较危险一点～当设定为 YES 时，表示每个建立的联机
  都会拥有一支 process 在负责，可以增加 vsftpd 的效能。不过，
  除非您的系统比较安全，而且硬件配备比较高，否则容易耗尽系统资源喔！
  一般建议设定为 NO 的啦！
tcp_wrappers=YES (NO)
  当然我们都习惯支持 TCP Wrappers 的啦！所以设定为 YES 吧！
xferlog_enable=YES (NO)
  当设定为 YES 时，使用者上传与下载档案都会被纪录起来。记录档案
  与下一个设定项目有关：
xferlog_file=/var/log/vsftpd.log
  如果上一个 xferlog_enable=YES 的话，这里就可以设定了！
  这个是登录档的档名啦！
xferlog_std_format=YES (NO)
  是否设定为 wu ftp 相同的登录档格式？！预设为 NO ，因为登录档会比较容易读！
  不过，如果您有使用 wu ftp 登录文件的分析软件，这里才需要设定为 YES
nopriv_user=nobody
  我们的 vsftpd 预设以 nobody 作为此一服务执行者的权限。因为 nobody 的权限
  相当的低，因此即使被入侵，入侵者仅能取得 nobody 的权限喔！
```

```
pam_service_name=vsftpd
  这个是 pam 模块的名称，我们放置在 /etc/pam.d/vsftpd 即是这个咚咚！
```

上面这些是相当常见的 vsftpd 的设定参数，还有很多参数我没有列出来，您可以使用 man 5 vsftpd.conf 查阅喔！不过，基本上上面这些参数已经够我们设定 vsftpd 啰！

最简单的 vsftpd.conf 设定

如果您很懒的去设定 vsftpd 的话，那么可以使用很简单的设定值来规划您的 FTP 服务器。底下就是 Red Hat 9 的预设 vsftpd 的设定值，您可以使用这样的设定值来启动您的 FTP 服务器即可。这样的设定值有几个用处：

- 任何在 /etc/vsftpd.ftpusers 里面的使用者账号均无法使用 vsftpd 喔！
- 开放 anonymous 与 实体用户 登入 vsftpd ；
- 实体用户登入主机时，可以跳至任何具有登入权限的目录当中(没有 chroot)；
- 使用 port 20 作为主动联机时的 ftp-data 传送埠口；
- 利用 /etc/hosts.allow(deny) 来管理登入权限；
- 当 Client 上传/下载档案时，该信息会记录在 /var/log/vsftpd.log 里面；
- 其它的设定均已默认值来规范(如被动式 port number 等等)。

```
[root@test root]# vi /etc/xinetd.d/vsftpd
service ftp
{
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/local/sbin/vsftpd
        server_args             = /etc/vsftpd.conf
# 上面这个请依照您的主机环境来设定！尤其是 server_args 请设定您的
# vsftpd.conf 所在目录的完整文件名(含目录名称)！
        log_on_success          += DURATION USERID
        log_on_failure          += USERID
        nice                    = 10
        disable                 = no
}
[root@test root]# vi /etc/vsftpd/vsftpd.conf  # (或 /etc/vsftpd.conf)
# 关于主机与安全性的设定
use_localtime=YES
dirmessage_enable=YES
connect_from_port_20=YES
```

```
xferlog_enable=YES
xferlog_std_format=YES
pam_service_name=vsftpd
tcp_wrappers=YES
# 关于 anonymous 的设定
anonymous_enable=YES
# 关于 real user 的设定
local_enable=YES
write_enable=YES
local_umask=022
userlist_enable=YES
# 以上设定值的意义请往前翻到 vsftpd.conf 设定值的意义 章节去察看!
[root@test root]# /etc/rc.d/init.d/xinetd restart
```

这样您的最简易的 FTP 服务器就已经设定完成了！简单的很吧！而且还相当的安全呢！

针对仅有开放实体用户登入的设定

好了，这里我们再使用其它的设定值来修正我们的 vsftpd.conf 这个设定档。因为开放 anonymous 毕竟不太安全，所以我们将 anonymous 的登入权限关闭，并且仅让 real user (实体用户) 登入我们的 vsftpd 时，要如何设定呢？我的要求如下：

- 使用台湾本地的时间而不是 GMT 时间;
- 所有在 /etc/passwd 里面出现的实体账号均能登入 vsftpd 主机;
- 但是系统账号 (如 root 等， UID 小于 500 的账号)均不能使用 vsftpd ;
- 而且由于 badbird 与 nogoodbird 这两个账号使用者比较不乖，我要让这两个使用者被关在自己的家目录当中(chroot);
- 并且限制数据的传输速度为 100 Kbytes/second;
- 当使用者进入 /home 这个目录时，显示：『一般使用者家目录』的字样在 Client 端的屏幕上;
- 使用者可以进行上传、下载以及修改档案等等动作。

```
1. 基础设定档
[root@test root]# vi /etc/vsftpd/vsftpd.conf (或 /etc/vsftpd.conf)
# 关于主机与安全性的设定
use_localtime=YES
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
pam_service_name=vsftpd
```

```
tcp_wrappers=YES
# 关于 anonymous 的设定
anonymous_enable=NO
# 关于 Real User 的设定
local_enable=YES
write_enable=YES
local_umask=022
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd.chroot_list
userlist_deny=YES
userlist_file=/etc/vsftpd.user_list
local_max_rate=100000
# 以上设定值的意义请往前翻到 vsftpd.conf 设定值的意义 章节去察看!

2. 限制实体用户在自己的家目录内 (chroot) 的设定档
[root@test root]# vi /etc/vsftpd.chroot_list
badbird
nogoodbird
# 没有写到这个档案内的其它用户，就可以离开自己的家目录，
# 而到其它目录里面去浏览了!

3. 以 PAM 模块限制某些账号无法登入主机的设定:
[root@test root]# vi /etc/pam.d/vsftpd
# 会发现这样的字句:
auth ..... file=/etc/vsftpd.ftpusers ....
# 那个 file=.. 后面接的文件名就是以 PAM 模块抵挡的账号内容了!
[root@test root]# vi /etc/vsftpd.ftpusers
# 底下列出的账号将无法使用 vsftpd 喔! 与 wu ftp 的 /etc/ftpusers 相同功能
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody

4. 以 userlist_file 抵挡某些账号的登入:
```

```
# 事实上，这个功能与上面的 PAM 功能相似啦！只是 PAM 是外挂的，而
# 这个设定是 vsftpd 预设提供的就是了！
[root@test root]# vi /etc/vsftpd.user_list
# 这个档案的设定与上面 /etc/vsftpd.ftpusers 相同即可！
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody

5. 设定进入目录时，显示的讯息：
[root@test root]# vi /home/.message
一般使用者家目录

6. 重新启动 xinetd 啰！
[root@test root]# /etc/rc.d/init.d/xinetd restart
```

上面的设定里面有很多重复的地方，比方说 /etc/vsftpd.ftpusers 与 /etc/vsftpd.user_list 就是重复的设定了！不过，这样是比较安全啦！因为 PAM 模块是外挂的程序，而另一个档案则是 vsftpd 提供的功能！但是请特别留意，因为很多使用者可能会一不小心仅修改了其中一个档案，另一个档案则忘记修订，会很麻烦喔！ ^_^！至于也是重点之一的 /etc/vsftpd.chroot_list 就可以将使用者限制在他们自己的家目录内了！设定上很容易吧！附带说明，上面的档名都与 vsftpd.conf 的设定有关！

看过了上面关于实体用户的设定之后，咦！那么如何让 root 可以登入 vsftpd 主机呢？！呵呵！就是将 /etc/vsftpd.ftpusers 与 /etc/vsftpd.user_list 这两个档案里面的 root 拿掉就可以啦！不过，本人可是不建议这么搞的喔！

针对仅有开放匿名使用者登入的设定

好了，上面一章节谈的是仅开放 Real User ，那么这个章节我们就来谈一谈，没有 Real user 仅有 anonymous 的登入说！我想要的功能是这样的：

- 使用台湾本地的时间，而非 GMT 时间；
- 仅开放 anonymous 的登入；
- 档案传输的速限为 30 Kbytes/second；
- 允许 anonymous 上传档案到 /var/ftp/upload 这个目录当中，并且允许 anonymous 建立目录；
- 数据连接的过程 (不是命令通道！) 只要超过 60 秒没有响应，就强制 Client 断线！
- 只要 anonymous 超过十分钟没有动作，就予以断线；
- 被动式连接的埠口为 65400 到 65420 这几个 port number 即可；
- 最大同时上线人数限制为 50 人，且同一 IP 来源最大联机数量为 5 人；
- 不许使用 ASCII 格式上传或下载！
- 不许以 linux.vbird.org 这个网址为 email address 的密码输入！

OK！这样要如何设定呢！？

```
1. 基础设定档
[root@test root]# vi /etc/vsftpd/vsftpd.conf (或 /etc/vsftpd.conf)
# 与主机与安全性有关的设定
use_localtime=YES
write_enable=YES
dirmessage_enable=YES
xferlog_enable=YES
xferlog_file=/var/log/vsftpd.log
data_connection_timeout=60
idle_session_timeout=600
max_clients=50
max_per_ip=5
ascii_upload_enable=NO
ascii_download_enable=NO
connect_from_port_20=YES
pasv_min_port=65400
pasv_max_port=65420
pam_service_name=vsftpd
tcp_wrappers=YES
nopriv_user=ftp
# 关于 anonymous 的设定
anonymous_enable=YES
anon_other_write_enable=YES
anon_mkdir_write_enable=YES
anon_upload_enable=YES
deny_email_enable=YES
banned_email_file=/etc/vsftpd.banned_emails
anon_max_rate=30000
# 关于 real user 的设定
```

```
local_enable=NO
# 以上设定值的意义请往前翻到 vsftpd.conf 设定值的意义 章节去察看!

2. 建立抵挡不当 email address 的档案
[root@test root]# vi /etc/vsftpd.banned_emails
linux.vbird.org
# 一行写一个 email 名称喔!

3. 建立可以上传的目录!
# 因为我们的 nopriv_user 设定为 ftp ,所以上传的目录拥有者为 ftp 喔
[root@test root]# mkdir -p /var/ftp/upload
[root@test root]# chown ftp /var/ftp/upload

4. 重新启动 xinetd 啰!
[root@test root]# /etc/rc.d/init.d/xinetd restart
```

经过上面的说明之后，您就可以很清楚的知道了 Real user 与 anonymous 在 vsftpd 当中的设定了，那么您就可以很轻易的就架设出一部 vsftpd 服务器了呢！赶紧尝试看看吧！ ^_^

Client 端的设定：

Client 端并没有什么好设定的地方，主要就是 ftp 的使用了，请参考 wu FTP 主机设定一节！

安全相关方面

有关安全性的设定方面，当然就是与登入以及防火墙有关啰！那就来谈一谈吧！

防火墙抵挡

要启用 vsftpd 自然就得要开放防火墙啰！所以您如果想要对 Internet 开放您的 FTP 服务器，就必须至少要有这一段 iptables 防火墙规则在您的规则列当中：

```
/sbin/iptables -A INPUT -p TCP -i eth0 --dport 21 -j ACCEPT
```

当然，您可以设定的更严密，请参考简易防火墙一文来设定 iptables 喔！此外，您的 TCP Wrappers 如果想要抵挡 192.168.1.2 这个 IP 来源的话，可以这样做：

```
[root@test root]# vi /etc/hosts.deny
vsftpd: 192.168.1.2
```

这个应该没有问题吧！ ^_^

Super daemon 的管理

Super daemon 可以用来管理 Client 端的登入权限呢，这个重点没有忘记吧！ ^_^ ！那么我们如何设定呢？！举个简单的例子好了：假设我们的 vsftpd 只允许同一个 IP 来源可以拥有五个登入权限，而同一时间最多可以有 200 个 vsftpd 的联机，当超过 200 个 vsftpd 的联机时，将在 Client 端的画面当中显示『很抱歉，服务器忙碌中』的字样，该如何设定？

```
[root@test root]# vi /etc/xinetd.d/vsftpd
# vsftpd is the secure FTP server.
service ftp
{
        disable                 = no
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/local/sbin/vsftpd
        server_args             = /etc/vsftpd.conf
# 上面这个 server 的设定请依照您的主机环境来设定！
# 至于 server_args 则请写入您的 vsftpd 的设定档完整档名即可！
        per_source              = 5      # 与同一 IP 的联机数目有关
        instances               = 200    # 同一时间最多的联机数目
        no_access               = 192.168.1.3
        banner_fail             = /etc/vsftpd.busy_banner
# 上面这个档案就是当主机忙碌中，则在 Client 端显示的内容！
        log_on_success          += PID HOST DURATION
        log_on_failure          += HOST
}

[root@test root]# vi /etc/vsftpd.busy.banner
421 很抱歉，服务器忙碌中！

[root@test root]# /etc/rc.d/init.d/xinetd restart
```

这样设定就可以啦！很简单的一个设定动作，就可以让您的 vsftpd 变的更安全一些喔！

问题解决方案

如果发生 vsftpd 的问题怎么办？！有几个可能的解决方案喔：

- 如果在 Client 端上面发现无法联机成功，请检查：
    1. iptables 防火墙的规则当中，是否开放了 client 端的 port 21 登入？
    2. 在 /etc/hosts.deny 当中，是否将 client 的登入权限挡住了？

3. 在 /etc/xinetd.d/vsftpd 当中，是否设定错误，导致 client 的登入权限被取消了？

- 如果 Client 已经连上 vsftpd 服务器，但是却显示『 XXX file can't be opend 』的字样，请检查：
    1. 最主要的原因还是在于在 vsftpd.conf 当中设定了检查某个档案，但是您却没有将该档案设定起来，所以，请检查 vsftpd.conf 里面所有设定的档案档名，使用 touch 这个指令将该档案建立起来即可！

- 如果 Client 已经连上 vsftpd 服务器，却无法使用某个账号登入，请检查：
    1. 在 vsftpd.conf 里面是否设定了使用 pam 模块来检验账号，以及利用 userlist_file 来管理账号？
    2. 请检查 /etc/vsftpd.ftpusers 以及 /etc/vsftpd.user_list 档案内是否将该账号写入了？！

- 如果 Client 无法上传档案，该如何是好？
    1. 最可能发生的原因就是在 vsftpd.conf 里面忘记加上这个设定『write_enable=YES』这个设定，请加入；
    2. 是否所要上传的目录『权限』不对，请以 chmod 或 chown 来修订；
    3. 是否 anonymous 的设定里面忘记加上了底下三个参数：
        - anon_other_write_enable=YES
        - anon_mkdir_write_enable=YES
        - anon_upload_enable=YES
    4. 是否因为设定了 email 抵挡机制，又将 email address 写入该档案中了！？请检查！
    5. 是否设定了不许 ASCII 格式传送，但 Client 端却以 ASCII 传送呢？请在 client 端以 binary 格式来传送档案！

上面是蛮常发现的错误，如果还是无法解决您的问题，请您务必分析一下这两个档案：/var/log/vsftpd.log 与 /var/log/messages ，里面有相当多的重要资料，可以提供给您进行除错喔！

对于 FTP 服务器软件选择的建议

在玩过了 Wu FTP, ProFTPD 以及 vsFTPD 之后，发现其实三个服务器各有优缺点啦！没有说哪一个特别的好，因为各有其利用的主机环境说！不过，其实我们还是有选择的思考方向啦。

- 考虑较为单纯的 FTP 设计，而且要求安全性：如果我们不要个别控制每个目录的流量、不必控制上传/下载比例、不必针对不同的实体用户或者是访客进行不同的权限设定，仅分 anonymous 与 real user 两种身份来让使用者登入主机的话，那么上上之选应该是 vsftpd 这个服务器才对！因为他设定上真的比较简单，而且在整体套件的设计上又比较安全喔！

- 多样化的设计，安全性要求亦不低：虽然 Wu FTP 与 Proftpd 可以达到的 FTP 服务器设定是一样的，不过 Proftpd 的设定又比 Wu FTP 简单一些，有点类似 Apache 的目录管理设定，此外， Proftpd 也可以达到控制上/下传比例、流量控制、针对不同的目录设定不一样的权限等设计，也比 Wu FTP 安全一些，所以，在多样化的 FTP 设计考虑下，可以选择 Proftpd 喔！

- 考虑主机的要求：某些主机本来就是含有 Wu FTP 或者是 vsftpd 等不同的 FTP 服务器软件，您又不想重新安装其它的服务器软件，那么使用您的 Linux distributions 所提供的 FTP 软件来架设您的 FTP 即可啊！不需要考虑其它的 FTP 软件咯。不过，请记得将您的 FTP 软件更新到最新版喔！避免被恶意攻击说！

事实上，也就是说，以 vsftpd 为主要的考虑依据，但是如果觉得 vsftpd 无法满足您的 FTP 服务器设计需求，就改以 Proftpd 来设计，最后，如果您也是懒人一族，使用 Wu FTP 就算了吧！^_^

参考资源：

- vsftpd 官方网站：http://vsftpd.beasts.org/
- man 5 vsftpd.conf

简易 Proxy Server 架设

最近更新日期：2004/11/12

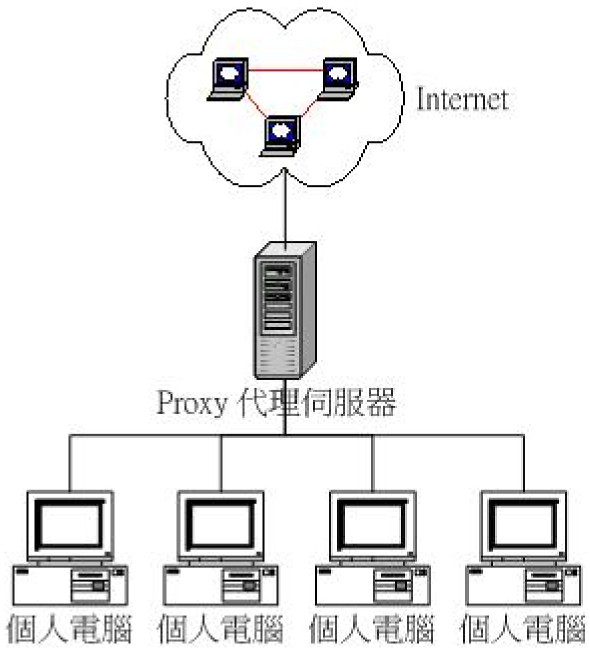
代理服务器的功能可多的呢！在中大型的企业当中，可以藉由单点对外的 Proxy 主机来达到『节省频宽』的目的，同时，也可以透过这样的 Proxy 架构来达成『高阶防火墙』的设定，这里的『高阶』指的是 OSI 七层协议里面比较高阶段的层级，那就是应用与表现层方面的防火墙啦！那如果对于小型的企业呢？这个 Proxy 也可以达到分流的作用，让不同的目标网站可以透过不同的上层 Proxy 来取得数据！啊！真是很不错的一个服务器啊！不过，这个 Proxy 服务器也是几个常见的服务器里面，硬件要求相对比较高的一个咚咚！因为 Proxy 要求的是『快速』，所以呢，呵呵！当然硬件等级的要求是相当的『蛮像一回事』的！^_^

原理：

代理服务器的原理其实很简单啦！就是以类似代理人的角度去取得使用者所需要的数据就是了！但是由于他的功用，使得我们可以透过代理服务器来额外的达成防火墙的功能！此外，也可以藉由代理服务器 (Proxy) 来达成节省频宽的目的，以及加快内部网络的 WWW 存取速度！总之，Proxy 对于大型的企业来说，实在是一个很不错的东西啊！

什么是代理服务器

在真实世界中，我们或许会帮忙家人去办理一些杂务吧！举个例子来说，例如缴费或者是申办提款卡等等的，那么由于你并不是『申请者本人』而是『代理人』的角色，因此有时候会需要秀出一些证件就是了。那么在网络上面的代理服务器是怎么回事呢？他就是 Proxy Server 啰！他最主要的功能就如同我们上面提的真实世界一样，Proxy 会帮 Client 端的用户去向目的地取得客户端所需要的数据。所以，当 Client 端指定代理服务器之后，您的所有相关要求(例如 WWW 的要求)就会通过代理服务器去捉取啰！整个代理服务器与客户端的相关性可以由下图约略看出一个端倪：



图一、代理服务器的示意图

在内部的计算机都是透过 Proxy 来向 Internet 求取数据的，这就是所谓的『代理服务器』啦！当然,上面的架构仅只是一个案例,还有相当多的非 Intranet 的 Proxy 架构,亦即是你的 PC 与 Proxy 均在 Internet 上面，但是您一样可以透过这个 Proxy 来帮您达到代理人身份的目的呢！

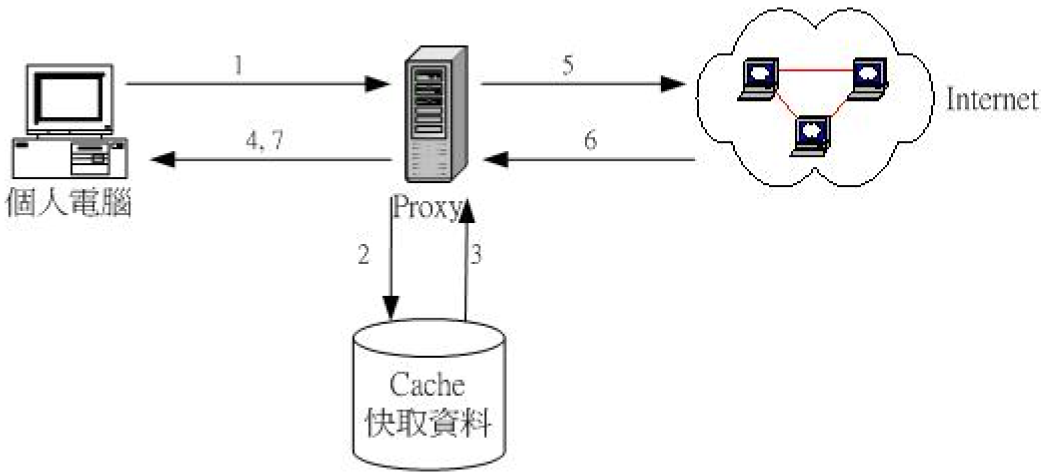
在 Proxy 与 Client 的相关性当中，您必需要了解的是：您向外部要求的资料事实上都是 Proxy 帮你取得的！怎么说呢？举个例子来说，假如我在我的浏览器 (假设是 Netscape 好了)设定了我们学校的代理服务器主机 proxy.ncku.edu.tw 做为我的 Proxy 好了，再假设我的 IP 是 140.116.44.125 ，那么当我想要取得奇摩网站的新闻信息时，事实上，都是 proxy.ncku.edu.tw 帮我去取得的，所以在奇摩的网站上面看到向他要求资料的人是谁呢？呵呵！当然就是

proxy.ncku.edu.tw 而不是我 140.116.44.125 啰！这样可以了解 Proxy 的功能了吗？

除了这点功能之外， Proxy 还有一个很棒的额外功能喔，那就是防火墙的功能！怎么说呢？看一下上面的图示，您可以发现一件事情，那就是 Client 端的个人计算机要连上 Internet 一定要经过 Proxy 服务器，并且，如果有人想要入侵你的系统时，由于你的 proxy 在最外部啊，所以攻击者就会攻击错方向，如此一来，不就比较安全！此外，由于整个 Intranet 对外都是经过 proxy ，也就是『单点对外』的情况，这种状态底下要来管理防火墙也是比较简单的喔！^_^

代理服务器的运作方式

了解的 Proxy 的功能之后，我们来谈一谈那么 Proxy 到底是怎样运作的呢？为何他会有『加快网络存取效率』的情况？这就必需要以底下的图示来说明了！



图二、代理服务器的运作方式流程图

当 Client 端设定了代理服务器之后，在 Client 端想要取得 Internet 上面的信息时，他是这样取得数据的 (注：那个 Cache 表示为 Proxy 主机的硬盘的意思)：

一、Proxy 有使用者预计要求的数据时(Step 1, 2, 3,4)：

1. Client 端向 Server 端发送一个数据需求封包；
2. Server 端接收之后，先比对这个封包的『来源』与预计要前往的『目标』网站是否为可接受？如果来源与目标都是合法的，或者说，来源与目标网站我们的 Proxy 都能帮忙取得数据时，那么 Server 端会预计开始替 Client 取得资料。这个步骤中比较重要的就是『比对政策』啦，有点像是认证的感觉啦；
3. Server 首先会到自己的硬盘里面，也就是所谓的 cache (快取) 查看一下有没有 Client 端所需要的数据，如果有的话，那就将数据直接送到 Client 端 (步骤 4) 而不经过向 Internet 要求数据的程序；

二、Proxy 没有使用者预计要求的数据时 (Step 1, 2, 3, 5, 6, 2, 3, 7)：

4. 在经过 1,2,3 查寻知道 cache 没有数据，或者数据过期之后， Proxy 会向 Internet 上面的目标网站要求数据；
5. 在将数据取回之后， proxy 会先将取得的数据『储存一份到 cache 当中』；
6. 最后才将数据传回给 Client 端使用。

整个 Proxy 的工作流程就是这个样子，所以，我们就可以知道的是， Proxy 对于 cache 的速度是很要求的，而这个 cache 就是硬盘啦！当然，硬盘容量必需要足够大，而且还要『足够快』才行！因为由上面的流程当中，我们不难发现， Cache 是一直被重复存取的一个地方喔！所以硬盘的好坏就差别很大啦！可以说他是影响一个 Proxy 效能好坏的关键点呢！

代理服务器的用途与优缺点

一般来说，代理服务器的用途主要有两个：

- WWW 网页代理人：最主要的用途当然就是做为网页资料取得代理人啰，也就是说， Proxy 可以帮我们取得 Internet 上面的 WWW 数据就是了！那么能不能取得其它非 WWW 的数据呢？那就不一定了，要看 Proxy 主机是否有设定该服务。一般来说， Proxy 主要还是针对 WWW 网页的代理取得；

- 做为 Intranet 的单点对外防火墙系统：就如同前面的图示， Proxy 如果架设在 Intranet 对外的连接点上面，那么他就可以被用来做为『应用层』阶段的防火墙了！而且，这个时候不需要设定 NAT 就可以让 Intranet 内部的私有 IP 的计算机连接上 Internet 了！这是因为您想要的数据是向 proxy 要求，所以真正去取得数据的人是『Proxy』而不是你的计算机啊！所以，只要 Proxy 可以接受私有 IP 的计算机要求，那这些私有 IP 的计算机就可以连上 WWW 啦！不过，也由于 Proxy 为一个 应用层 阶段的防火墙系统，所以，他并无法进行较低阶的封包过滤！因此，在内部计算机想要透过 Proxy 来取得邮件、或者是其它的服务，呵呵！那就比较麻烦，简直就是麻烦的多啦！

由于 Proxy 的这种特性，让他很常被使用于大型的企业内部，因为可以达到杜绝内部人员上班时使用非 WWW 以外的网络服务，而且还可以监测使用者的资料要求流向与流量呢！很不错吧！ ^_^ 好了，接下来我们来谈一谈 Proxy 主要的优缺点吧：

- 快速的存取动作：一般来说， Proxy 主机的频宽以及硬件配备会比较高档！所以 Proxy 最大的优点就是可以提供客户端较为快速的浏览！咦！但是我们向 Proxy 要求数据的时候，Proxy 不是会自行再储存一份吗？这样不是会多花很多时间？是这样没有错，但是，换一个角度来想，如果在第一位使用者要求过 A 数据后，由于 Proxy 就会自动放一放 A 数据在 Cache 当中，之后的所有使用者只要是重复要求这个 A 数据， Proxy 可以立刻将资料传给使用者，您瞧！这样这个 Client 等于是直接向 Proxy 取得这份 A 数据了！

是否更加的快速！这是因为 Proxy 就在您的 Intranet 之内，传输速度可是相当快的！这也就是说：如果您要设定代理服务器的时候，一定要找距离我们的机器最近的那一部，否则就没有达到代理服务器的功用了！通常快速的存取动作最明显的大概是连去国外的网站了！这里要强烈的建议，如果你需要连上国外的网页，请一定使用代理服务器，因为不但可以节省频宽，并且速度上会快上很多很多（ 例如美国环保署，EPA 网站！ ）

- 降低网络的负荷：由于我们是向代理服务器要求数据，如果代理服务器内刚好有你要的数据，将会直接传给你，则你的要求将不会到真实的那一个网页去（ 除非你在 IE 内按下『重新整理』这个按钮 ），而如果没有你要求的数据，那他也会去捉一份你要的数据给你，并存下来，以后如果有与你相同需要的用户，那他就可以直接传送给用户，如此当可降低网络的负荷！(也就是上面图二的 step 1, 2, 3, 4)

- 资料分流：由于各家 ISP 对于不同国家的频宽是有差异的，因此，假设如果您要去美国时使用 Proxy1 速度较快，而 Proxy2 则是去日本比较快，至于台湾本地则 Proxy3 较快，如此一来，我们可以透过设定将不同目标的代理服务器分开来，以达到分流的目的！则你的网域中将可以达到很好的分流效果，网络『感觉上』会比较快速喔！

- 提供防火墙内部的计算机连上 Internet：这个是一般企业比较常用的情况！由于企业内部害怕被黑客侵入，通常会设立一些比较严密的防火墙，然而如此一来公司内部的计算机可能面临无法连上 Internet 的窘境，那使用 proxy 让你的内部计算机可以透过这一架主机的代理服务而取得 Internet 上的信息，就是一个很好的方法啦！

- 多层次的管道（ 上层代理服务器 ）：代理服务器可以提供多重的管道设定，例如，当你需要国内的数据时，代理服务器将直接去捉取，而需要国外的数据时，才连到上一层的代理服务器!如此将可达到你的需求（ 而不用常常在你的 IE 等浏览器上更改所需的代理服务器 ），这个部分我们在底下还会进行额外的说明。

有利就有弊，当然 Proxy 也不是万能的天神～他有什么可能潜藏的缺点呢？

- 容易为 Intranet 的内部人员滥用：因为 Proxy 是对内部的计算机捉取数据（ 当然也可以对 Internet 上面的使用者捉取数据啦！ ），而且在 Internet 上面看到的实际上是你的 Proxy 在捉资料喔！如果你的使用者大量的以浏览器下载 A 图啊，还是透过你的 Proxy 干坏事啦，这样一来可就累了～因为实在不容易轻易的管理！所以，为了杜绝这个状况，强烈的建议多加安装登录档案分析的软件，在管理上面会轻松很多喔！

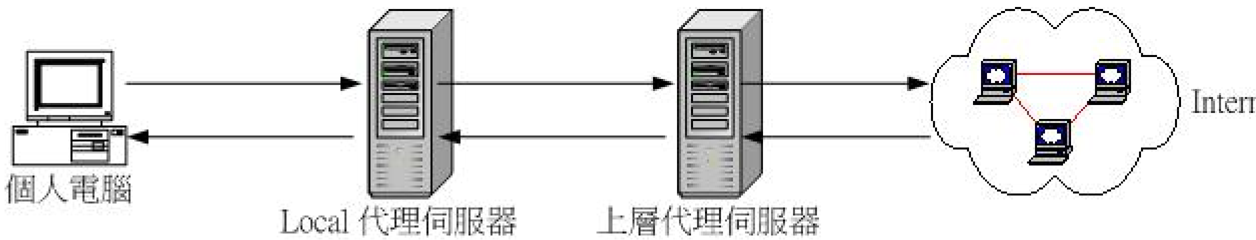
- 需要较高超的设定技巧与除错程序：在鸟哥设定过的 Server 当中， Proxy 算是比较不容易设定好『效能』的一个服务器了！由上面的传输过程中，您不难发现 Proxy 的 Cache 与他的『上层代理服务器』的关系是很紧密的，万一设定错误的话，很有可能反而让您的 Proxy 拖垮 WWW 的浏览速度！最严重的是造成无法联机 (在上层 Proxy 与您的 Proxy 之间构成 loop 而跑不出去！)，因此，这对于管理员来说是比较困扰的一件事。

- 可能会取得旧的错误数据：由前面的 Proxy 运作过程当中不难发现，Client 端向 Server 端求取资料时，Server 会先向自己的 cache 查寻，如果有该索求数据，就立即将数据送给 Client 。现在假设个例子来说明，万一我的网页三天两头改变一次，那么那个 cache 事实上并没有天天更新啊！这个时候， Client 端所取得的数据就有可能是网页修改之前的旧数据咯！所以，使用者得常常按下『更新』才能取得新的资料啊！

总之， Proxy 的优点是很多的，但是缺点却需要网管人员的操心啊！

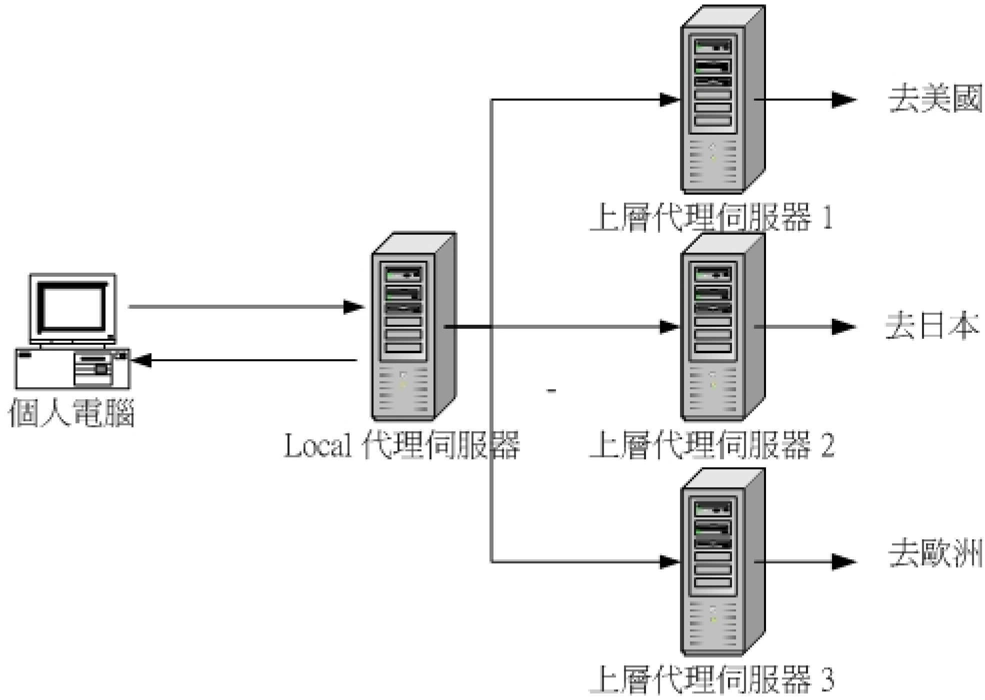
什么是上层代理服务器？哪里有上层代理服务器

什么是上层代理服务器：
好了，上面提到过所谓的『上层代理服务器』这又是什么咚咚？事实上，上层代理服务器就是一个 Proxy 啦，只是，我们自己设定的这个区域 Proxy 会将自己当作 Client 而去要求另外一个 Proxy 求取数据来给我们的使用者就是了！整个流程图可以这样看：



图三、上层 Proxy 示意图

就是我们的 Local proxy 并不会主动的去捉数据，而是透过『上层代理服务器』去向 Internet 要求数据！这样有什么好处呢？刚刚上面也曾经提过了，由于这些上层代理服务器才是真正对外频宽最大的几部机器之一，所以透过他来要求数据一定又比我们的 Local proxy 还要来的快啊！所以我当然会喜欢设定上层 Proxy 噜！这个现象最常发生在对国外的联机上面，有没有设定 Proxy 差异是相当大的呢！上层代理服务器除了频宽更大之外，还有没有什么好处啊？当然有，最大的好处就是达到分流的效应！例如下图的说明：



图四、上层代理服务器的分流动作示意图

我总共设定了三个上层代理服务器，由于这三个代理服务器对外的速度都不相同，所以，当我要去美国时，就以 Proxy1 来要求资料，要连欧洲就以 Proxy3 ，至于要连日本，就以 Proxy 2 来要求我所需要的数据，如此一来，呵呵！可以让我的 Proxy 达到最佳的效能喔！很不错吧！ ^_^！所以上层代理服务器是很重要的呢！

哪里有上层代理服务器？

目前有哪些流量大、然后又开放出来的 Proxy 呢？我这里举几个网页给大家参考参考：

- SeedNet 的代理服务器(http://service.seed.net.tw/dial/server.shtml)；
- Hinet 的代理服务器(http://www.hinet.net/support/new_adsl04.htm)；
- 一些台湾学术网络的代理服务器 (http://turtle.ee.ncku.edu.tw/~tung/proxy/proxylst.html)。

由前面的介绍中，我们不难发现 Proxy 有可能会被 Client 端过度的滥用，同时也有可能会被拿来为非作歹啊！所以，目前绝大部分的 Proxy 已经『停止对外开放』了，仅针对自己的网域内的 Client 提供 Proxy 的服务而已～因此，如果您要自行设定 Proxy 的时候，请记得去您当初申请网络的 ISP（ 如果是学术单位，就到上面介绍的学术网络查看看即可！） 查寻一下，才能比较有效的设定好您的主机喔！因为设定错误的话，呵呵！上层 Proxy 根本不提供服务，或者是上层 Proxy 的效能并不好，那个时候您的 Proxy 也会连带的受到很大的影响啊！慎选！慎选！

我是否一定要设定 Proxy ？

话又说回来，到底我应不应该设定 Proxy 呢？还是得由理论与实际上的状态来进行说明。事实上，

我们的 Proxy 感觉上会加快传输的速度，主要的因素是来自于 cache 已经记录了一份数据了，所以 Client 端取得的其实是这一份数据，而不是真的来自于 Internet 上面的实时数据！这样的好处前面提过了，就是可以增加内部网络传输的效能啊！但是，这要在一个前提之下，就是我的使用者很多时，那么由于来自四面八方的人会四处去求取资料，让我的 Cache 拥有较大的数据库，那么内部传输的速度自然就会有所帮助！所以，要架设 Proxy 的情况可以是：

- 我的 Client 端用户不少，而且大部分仅需要 WWW 这个网络服务而已；
- 我的 Proxy 还兼做防火墙的任务；
- 我的 Client 端常常需要联机到传输速度很慢的网站，例如国外的网站；
- 我的 Client 端常常浏览的网站是『静态』网站，而不是动态网站(例如讨论区的 PHP)。

相反的来说，要是 (1)我的 Client 端很少，那么每次上去 WWW 都是求取新的资料，有没有 Proxy 反而看不出效益～此外，(2)Proxy 由于属于应用层了，对于 Internet 的规划上弹性较不足！不像 NAT 主机可以进行很多的功能！(3)我常常上的网站是类似讨论区那种一日多变的网站，在这样的情况下，实在是没有必要架设 Proxy 的！

但是，如果对于学校单位那原本频宽就不足的环境中，架设 Proxy 来让校内的网络速度提升，呵呵！就是有那个必要性的啦！所以要不要架设 Proxy 呢？请好好的依据您的环境来考虑喔！

所需要的硬件要求与最佳硬件配置方式

假设您一定需要架设 Proxy ，那么到底什么样的配备是必需的呢？我们刚刚提到 Proxy 对于硬件的要求相当的高！因为我们架设 Proxy 的目的就是希望能够加快网络的传输效能嘛！因此，虽然 Proxy Server 几乎在任何的 Linux 系统上面都能跑 (例如我的 P 133 MMX) ，但是您的 Proxy Server 最好还是能有以下的硬件等级：

- CPU 最好能够 P III 550 以上等级；
- RAM 最好能够大于 512 MB ，这也是很重要的一个硬件参数！
- Hard Disk 最好能用 SCSI 接口的，因为速度与稳定度都比较好！如果不能的话，那么 IDE 接口的硬盘由于目前速度也越来越快，所以使用 IDE 也没有问题，但是最好是『多颗硬盘』的架构，例如我总共需要 30 GB 的硬盘空间，那么最好是 10 GB 的硬盘三颗这样的架构较佳！为什么呢？由于 cache 对于 Proxy 的重要性相当的大，所以 cache 读取速度越快的话，代表 Proxy 的效能也会越好！那由于 Proxy 对每笔资料写入 cache 时，是『平均分摊在各个 cache 的目录中 』，所以当然硬盘数越多越好啰！例如我原本有 10 MB 的数据要写入 cache ，那么这 10MB 写入同一颗硬盘快？还是 10MB 被分散写入三颗硬盘，因此每一颗硬盘仅记录 3.3MB ，那一个快？当然是三颗硬盘的架构会比较快，因为我有三个磁头在帮我写入数据嘛！^_^！请注意喔！这里我们说的是『多颗硬盘』而不是『多个 Partition 』喔！因为如果我将 30GB 的硬盘切割成三块 partition 的状态下，由于还是只有一个磁头啊，所以写入的速度差异不会很大！这里要特别留意与了解呢！

- 网络卡与网络周边最好使用 GBytes 的网络卡，当然啦，一般的公司行号应该不需要用到这样的网络卡才是！我这里指的是较高档的配备啦！

事实上，最重要的还是 RAM 与 HD 这两样，当然，网络接口也绝对不能忽视就是了！针对硬盘来说，最好使用 SCSI 这个稳定的接口，当然，使用 IDE 也是不错啦！但是要注意保养就是了。此外，就如同上面提到的，在硬盘的架构上是相当的重要的！一般来说，使用磁盘阵列应该是不错的想法，如果没有办法的话，使用多颗硬盘取代单颗硬盘的架构，在效能上也会有不错的显著提升呢！

既然硬盘这么重要，我们也约略谈一谈硬盘的基础规划应该有哪些需要注意的呢？

- 最好在架设 Proxy 时，将整体主机的规划做好，并且让 Proxy 主机的服务简单一点，就是仅负责 Proxy 就好了！
- 每颗硬盘的容量不需要太大，大约在 9 GB 以内即可，此外，最好将硬盘分割一下，一块 Partition 差不多在 2~4 GB 之间即可，因为切太大的话数据的搜寻耗费时间较长，但是 Partition 太小又可能造成空间的浪费～所以差不多的大小就限制在 2~4 GB 吧！
- 我们刚刚上面有提过，cache 是放置在某个目录下的，而最好一个目录底下就是独立的一个 partition 。此外，由于 cache 所在的硬盘常常会有数据的存取，因此可能此一硬盘的损耗率会比较大，所以这个 cache 所在的硬盘最好不要跟重要数据文件，例如 /, /etc, /usr, /home 等等重要的系统档案放在一起，以免危险啊！
- 也由于 cache 所在的硬盘数据存取太密集了，所以，硬盘的选择上面需要 (1)转速不能太低； (2)磁头的机械臂需要可以忍受频繁的动作； (3)发热量不可太大，或者可以考虑加装硬盘用风扇。

代理服务器与 NAT 主机的差异

或许您已经发现了一件事，那就是：在内部局域网络使用私有 IP 的 Client 端不论透过 Proxy 或者 NAT 均可以直接取得 WWW 这个 Internet 的服务，那么 NAT 与 Proxy 有没有什么不同的地方啊？他们不都是可以让内部的 Client 连接出去吗？其实这两个玩意儿差异性是『相当大』的：

- NAT 是一个利用 TCP/IP 的 packet filter (封包过滤机制)来进行封包处理的一个机制，所以他是『直接分析 TCP/IP 』，所以在设定防火墙的时候，他的弹性比较高，只要能过 NAT 这一关，那么大部分的网络服务都可以使用，因为 TCP/IP 是比较底层的协议啊！要知道的是，TCP/IP 上头还有 port 、还有 IP 等等的信息，单是 port 就可以让我们使用很多的不同的协议了！例如 port 20, 21 是 FTP 啊， 80 是 WWW 啊等等的！所以 NAT 能做的事情事很多的！

- 至于 Proxy 就不一样了， Proxy 主要透过类似 Squid 这一类的软件来达成的一个服务，基本上，一般来说他是透过 port 3128 来进行数据的监听与传输，单是看到这个 port 3128 就应该要晓得他仅是一个 daemon 而已。Proxy 已经是应用层这个阶段的网络项目了，所以他并没有去分析 TCP 的封包，只要 Client 来源合乎他的需求 (例如 IP 是被支持的) ，那么他将透过 daemon 的功能帮使用者达成使用者所想要的任务！所以说，能不能做某些事情，与 Proxy 服务器上面负责的那个 daemon 是有关的！万一 daemon 无法进行 FTP 数据的取得，那么您再怎么努力的尝试上网也是枉然的啊！

这样说有没有比较有点概念了呢？NAT 是由较底层的网络去进行分析的工作，至于通过 NAT 的封包是干嘛用的， NAT 不去管他！至于 proxy 则主要是由一个 daemon 的功能达成的，所以必需要符合该 daemon 的需求，才能达到某些功能！

谈完了这些基本的原理之后，我们可得来玩一玩 Proxy 了吧？！事实上，目前有很多的 Proxy 软件，例如 apache 也有提供 proxy 模块的功能喔！但是，最好不要用 Apache 当作您的 Proxy server ，因为.....效能真的太差了～会让您的网络停顿的更厉害～目前 Proxy 的服务器软件当中，以 squid 这个咚咚最出名，会出名的原因是因为他的效能高！真的很不错的一套软件，所以底下我们就针对 squid 来做说明吧！

套件安装

我们的 Proxy 服务器软件选择 squid 这个效能很高的套件来安装，目前 (2003/03) Squid 已经出到了 2.5 版了，您可以到官方网站上面下载，或者是到中山大学的 FTP 网站上面下载，底下提供一下联系的网址：

- 官方网站：http://www.squid-cache.org/Versions/v2/2.5/
- 中山大学：http://ftp.nsysu.edu.tw/Unix/Proxy/squid/source/STABLE/

同样的， squid 也有两种主要的安装模式，分别是 RPM 版本与 Tarball 版本，不过，由于我们会加入一些不同的参数设定值，所以预设的 RPM 档案并无法满足我们的需求，除非使用 SRPM 来进行重新 configuration 的动作，否则比较不能让我们满意啦!因此，习惯上我们都是以 Tarball 的方式来进行编译、设定与安装，所以底下鸟哥会比较偏重以 Tarball 的安装来进行介绍，如果有兴趣的话，可以尝试以 SRPM 进行修订的工作喔！

使用 RPM 方式安装 squid

一般来说，使用您的 Linux distribution 提供的 Squid 也就够了，但是，就如同上面提到的，可能会有一些设定值您无法自由自在的设定，不过。不过，无论如何，以 RPM 来安装是最简便的啦！那么就请拿出您的原版光盘，将他 Mount 上来，查询一下是否有 squid 字样的文件名称？没错，就将他安装上去吧！目前主要的几个 distribution 都有提供这个套件，所以应该可以很快的找到这个套件才是！怎么使用 RPM 呢？没这么难吧！

```
[root@test root]# rpm -ivh squidxxxxxxx
```

一再地强调， RPM 是粉重要的，请好好的使用他吧！ ^_^！这个指令就安装完毕啰！只是要注意的是，由于 squid 2.2 版以前的设定与 2.4 版以后的设定差异性很大，所以请特别留意您的 squid 版本，如果是使用旧的 Linux distribution 的使用者，例如 Red Hat 6.xx 版本，或者是 Mandrake 7.xx 版本，那么就不要再以 RPM 来升级了！直接使用底下的 Tarball 吧！反正才安装一个档案，还不会太难啦！

使用 Tarball 方式安装 squid

一般来说，我还真是蛮喜欢中山大学的 FTP 站（ 怪了，好像一直在帮人家打广告！ ^_^ ）没办法，我们南部人嘛！当然是南部的 FTP 站比较亲切啰！这个时候，您可以直接在 Linux 底下使用 wget 来取得我们所需要的 squid 喔！目前我取得的版本是 2.5.STABLE2 版本，取得的方法如下：(注：您也可以到鸟哥的私房菜下载 http://linux.vbird.org/download)

```
[root@test root]#  wget  \
http://ftp.nsysu.edu.tw/Unix/Proxy/squid/source/STABLE/squid-2.5.STABLE2.tar.gz
```

好了，这个档案应该就是 /root/squid-2.5.STABLE2.tar.gz 啰！那么就开始来给他安装吧！过程很简单啦，重要的地方只有在下达 ./configure 的地方需要很多的额外参数支持就是了！

```
0. 解压缩：
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/squid-2.5.STABLE2.tar.gz
# ....(略)....会产生一个 squid-2.5.STABLE2 的目录
[root@test src]# cd squid-2.5.STABLE2

1. 开始设定参数：
[root@test squid-2.5.STABLE2]# export CFLAGES='-O2 -mcpu=i586'
# 这是一个额外的参数啦！因为我们使用的是 Linux ，而我们的 GNU gcc
# 有针对每种不同的 CPU 来进行套件的最佳化编译！所以啦，就加入我们的
# CPU 型号吧！因为我的 CPU 是 P-166 ，反正是个小案例啦！所以才会是 i586，
# 您的 CPU 只要超过赛扬等级以上，就会是 i686 啰！此外，除了
# i386, i486, i586, i686 还有 pentium, pentium3, pentium4, athlon,
# athlon-tbird, athlon-4, athlon-x, athlon-mp, k6, k6-2, k6-3 等等！
# 如果不确定您的 CPU 那么就用 ix86 之类的方式来命名吧！不过，即使没有
# 写入这个参数也无所谓啦！至于那个 O2 是最佳化参数啦！

[root@test squid-2.5.STABLE2]# ./configure --prefix=/usr/local/squid  \
> --enable-gnuregex --enable-async-io=80 --enable-icmp \
> --enable-kill-parent-hack --enable-snmp  \
> --disable-ident-lookups --enable-cahce-digests \
```

```
> --enable-err-language="Traditional_Chinese"  \
> --enable-poll --enable-linux-netfilter
# 这个咚咚就有趣啦！因为实在有相当多的参数可以使用，你可以使用
# ./configure --help 来察看可以使用的许多参数啊！稍微解释一下各个设定值：
--prefix=/usr/local/squid：未来程序编译完成后放置的安装目录；
--enable-gnuregex：使用 GNU 提供的正规表示法的原则来进行编译，请注意，
    因为在 Proxy 未来的规划当中，很可能会动用到正规表示法的方式来
    抵挡一些恶意的网站，所以这里应该要加入这个参数的！
--enable-async-io=80：这个项目主要在控制一些输出入的组件，使用这个项目
    可以让您的 Proxy 效能提升很多喔，因为是异步输出 (async) 的模式啊！
    后面接的数值是可以变动的，如果您的网站配备很高档，可以尝试将这个数字
    提升到 160 以上，如果是小网站的话，那么可以考虑将他降低至 40 左右。
--enable-icmp：要不要支援 ICMP 啊！当然是要的！
--enbale-kill-parent-hack：在我关掉 squid 的时候，要不要连同
    parent process 一起关掉，当然也是要的啦！
--enable-snmp：这个与制图的 MRTG 比较有关啦！如果没有用到的话，
    可以考虑将这个项目拿掉也没有关系！
--enable-cache-digests：这个项目很重要的啦，我们在底下再进行说明。
--enable-err-language="Traditional_Chinese"：不需要写了吧？
    只要有任何的错误讯息，网页上面显示的语系会是中文喔！
--enable-poll ：可以提升效能；
--enable-linux-netfilter：可以增加通透式 Proxy 的设定！后面再提啦！

2. 开始编译以及 Install 啰！
[root@test root]# make && make install

3. 开始设定其它的相关参数
[root@test root]# vi /etc/man.config
# 在这个档案当中新加入一行：
MANPATH /usr/local/squid/man   # 与 Squid 有关的 man page
# 可以让 squid 提供的说明文件让 man 指令可以查到！
```

就这样几个简单的步骤就将 squid 给他安装完毕啦！很快速吧！所以我说，使用 Tarball 来安装 squid 其实是很快的，不用太担心啰！

Server 端设定：

终于来到了主机端的设定项目啦！以下我们会分门别类的介绍各个主要的参数值，这些参数不见得适合您的 Proxy 环境，所以使用的时候敬请特别小心喔！每个项目都要好好的了解一下呐！另外，底下我主要是以 Tarball 安装的目录为主要的介绍状态，如果您是以 RPM 来安装的话，那么所有的档案原理与设定还是一样的，只不过档案存放的路径就不太一样就是了！请以 RPM 的指令或者是 locate 与 find 的方式找出来您的设定档吧！

---

squid 的结构

刚刚安装的目录其实是在 /usr/local/squid 这个目录下，而这个目录又分为几个主要的子目录，分别为：

- bin/ ：放置主要的 squid 执行 scripts 的目录，重要的是 RunCache 那个档案；
- etc/ ：几乎所有的 squid 设定档都在这里；
- libexec/ ：一些函式库；
- man/ ：就是一些在线文件查寻档啦！
- sbin/ ：重要的就是那个 squid 的执行档！
- share/ ：一些错误讯息代码表示档案，以及一些小图标放置的目录；
- var/ ：预设是放置 log file 的，不过我不喜欢放在这里，这点等一下我们会修改的！

那么主要的设定档有哪些呢？其实可以说只有两个啦：

- /usr/local/squid/etc/squid.conf ：这个是主要的设定档，所有的 squid 所需要的设定都是放置在这个档案当中的！鸟哥底下提到的种种设定方法几乎都是这个档案里面的说明喔！所以，如果您英文不错，那么就直接看一下这个档案就知道如何设定 squid 啦！
- /usr/local/squid/etc/mime.conf ：这个档案则是在设定 squid 所支持的 Internet 上面的档案格式，就是所谓的 mime 格式啰！一般来说，这个档案的预设内容已经能够符合我们的需求了，所以不需要更动他，除非您很清楚的知道您所需要额外支持的 mime 档案格式。

而执行档其实只有一个，那就是 squid 啦！不过， Squid 这个套件额外的提供了两个可执行的 scripts 来帮助大家执行 squid ，那就是在 bin/ 里面的 RunAccel 与 RunCache ：

- /usr/local/squid/sbin/squid ：就是我们说的 squid 的执行档啰！要知道这个指令的参数吗？就使用『 ./squid --help 』就能知道有什么参数啦！
- /usr/local/squid/bin/RunCache ：这个是主要的执行 squid 的一支简单的 script ，主要是利用 squid.conf 设定档案的内容来启用 squid 喔！
- /usr/local/squid/bin/RunAccel ：如果您的 WWW 服务也想要透过 Squid 来进行『加速』的话，那就可以使用 RunAccel 来取代 RunCache 了，不过，我通常还是使用 RunCache 而已！

---

squid 的 process owner 与 cache directory owner

我们前面的原理部分稍微提过，squid 主要是以 daemon 提供的 Proxy 功能，而这个 daemon 最大的功能就是将 Internet 上面捉取的数据给他放入 Cache 目录当中啦！而由于 daemon 会产生

一些 processes (程序)，这些程序都会有 Owner 以及 Group 。这样晓得我要讲什么了吧？呵呵！没错，那个放置 cache 的目录 (底下简称 cache dir) 的拥有者以及拥有群组就必需要与 squid 产生的 process 的拥有者与群组相同才行!而为了保险起见,通常 squid 不会以 root 来启动，最好是以 nobody 或者是一些权限比较低的系统账号来启动他！假设我们的 squid 这个 daemon 是由 nobody 所启动的好了，而假设我的 cache dir 是放置在 /var/spool/squid 下面，则这个 /var/spool/squid 的 Owner 与 group 就必需要是 nobody, nobody 才行！这个很重要喔！因为大部分无法启用 squid 的朋友都是这个动作没有搞正确的原因啊！

squid 的 Owner 与 Group 是在 squid.conf 里面设定的，而至于 cache dir 则是需要我们手动来设定好他的权限呢！

最简易的 squid 设定方法

设定 squid 仅要修改一个档案而已，那就是 /usr/local/squid/etc/squid.conf 啰(RPM 版本就不相同啰！请自行找出来吧！)！请注意，在这个例子当中，我们并没有介绍高档配备的设定，仅只是列出重要的设定项目还有一些观念，您所想要的设定必需要视您的主机规划而定，例如 cache dir 每个人所放置的目录都不相同啊，所以直接拿我的设定来启动时，可能会完蛋啊！请注意需要修改成您所想要的样式才行！好了，在这个小节当中，我们仅列出来几个一定要设定的参数，至于更进阶的参数将留待后面分别介绍。

另外要额外提醒的是，在 squid.conf 这个档案当中，预设的情况下是『除了本机可以使用 squid 的少部分功能外，其它所有的项目都没有被启动』，所以您必需以 vi 的搜寻功能找到下列的设定项目后，将批注符号 (#) 拿掉，或者是自行输入底下的设定才行喔！不啰唆，马上进行吧！

```
[root@test root]# cd /usr/local/squid/etc
[root@test root]# vi squid.conf
# 1. 关于网络的参数设定部分
#    在这个部分当中，最重要的就是启用 squid 这个 daemon 的 port 了！
#    在预设的情况下，公认的标准 proxy port 为 3128 ，至于被查询封包
#    观察的则是 3130 这个 port，这里我们分别启用这两个 port ！如果您的
#    Proxy 还有帮人家代理 https 这个由 SSL 协议启用的 port ，那么还需要
#    启动 https_port ，但是我们这里不谈论 SSL 啦！太危险了～
http_port 3128
icp_port  3130

# 2. 设定快取目录 ( Cache dir ) 的大小与记录档案所在的目录
#    这个设定是重要到爆的地方，一定得设定正确才行啊！上面我们不是
#    提过关于硬盘与目录吗？好了，现在这样假设：我有两块 partition ，
#    这两块 partition 分别挂载在 /usr/local/squid/var/cache1
#    以及 /usr/local/squid/var/cache2 这两个区域，此外，两块
#    partition 一块为 1GB (cache1) 另一块为 2 GB (cache2) ，则设定为：
#    <cache_dir> <aufs|ufs> <目录所在> <MBytes 大小> <dir1> <dir2>
#    那个 aufs 只有在编译的时候加入 --enable-async-io 那个项目才有支持，
```

```
#     至于目录所在地与所占用的磁盘大小则请视您的主机情况而定，
#     而后面 dir1, dir2 则是两个次目录的大小，通常 16 256 或 64 64 皆可，
#     一般来说，数字最好是 16 的倍数，据说效能会比较好啦！
#     注意 1：下面两行需要『视您的主机环境而定！』不要照抄！
#     注意 2：在底下的例子中，我的两块 partition 已经 mount 上该目录了！
#             这也就是说，底下的两个目录是『已经存在的！』
cache_dir aufs /usr/local/squid/var/cache1 1000 16 256
cache_dir aufs /usr/local/squid/var/cache2 2000 16 256
#     底下则是关于记录文件的放置目录与文件名！
cache_access_log /usr/local/squid/var/logs/access.log
cache_log /usr/local/squid/var/logs/cache.log
cache_store_log /usr/local/squid/var/logs/store.log
pid_filename /usr/local/squid/var/logs/squid.pid

# 3. 关闭认证机制
#     不晓得为什么，这一版的 squid 会自动的加入认证机制，请找到底下
#     几行，将他 mark 起来！
#auth_param basic children 5
#auth_param basic realm Squid proxy-caching web server
#auth_param basic credentialsttl 2 hours

# 4. 提供 squid 服务
#     预设的情况下，仅有本机可以使用 squid ，我们先将所有的权限开放
#     然后在一个一个的关闭啰！先找到底下这一行：
http_access deny all
#     将他改成
http_access allow all

# 5. 设定 squid 的拥有者与系统管理员信箱：
#     就是刚刚我们上一小节提到的 squid 的拥有者，请注意，这个
#     user 与 group 必需要在 /etc/passwd 及 /etc/group 里面存在方可成功！
#     我这里以权限最小的 nobody, nogroup 来做为范例，您也可以自行设定！
#     另外，cache_mgr 则是 squid 管理员的信箱，当 squid 发生问题时，
#     屏幕上就会出现这个信箱给使用者联系管理员之用！
cache_effective_user   nobody
cache_effective_group  nogroup  # 您也可以改成 nobody ！
cache_mgr youraccount@your.e.mail

# 6. 变更目录权限：
#     在预设的情况下，我们主要的纪录信息都写入 /usr/local/squid/var 里面，
#     所以这个时候需要将这个目录的权限改变成为 nobody 与 nogroup 所有！
#     当然啰，如果您的 cache_dir 不在这个目录当中，那么还需要额外自行建立，
#     例如我的 cache_dir 万一是在 /proxy1 与 /proxy2 时，那么我就必需要：
#     chown -R nobody:nogroup /proxy1
```

```
#     chown -R nobody:nogroup /proxy2
#     关于权限的问题是很重要的！请不要忘记了！
[root@test root]# chown -R nobody:nogroup /usr/local/squid/var

# 7. 开始启动 squid：
#     启动 squid 来试看看吧！不过，首先我们必需要建立快取目录的格式
#     才行，此外，由于我们想要以 nobody 来启动 squid ，所以你需要这样：
[root@test root]# /usr/local/squid/sbin/squid -z # 建立 cache_dir
[root@test root]# su nobody -c "/usr/local/squid/bin/RunCache &"

# 8. 查看是否真的启动了 squid 了？
[root@test root]# netstat -tln | grep 3128
tcp        0      0 0.0.0.0:3128       0.0.0.0:*          LISTEN

# 9. 重新读取设定档 squid.conf 的方法：
[root@test root]# /usr/local/squid/sbin/squid -k reconfigure
```

在上面的设定中，重要的地方在于：

- 你的硬盘规划与 cache_dir 的设定是否吻合：请注意，除非您是『架设着玩的，纯粹锻炼功力』的角度来设定 Proxy ，那么可以直接以预设的设定来搞定您的 cache_dir ，不然的话，前面我们提过，这个 Cache dir 可是影响 Proxy 效能的相当重要的因素之一，因此，千万不可大意啊！

- 关于权限的大问题：由于我们常常提倡『不要以 root 来启动 daemon 』，所以这个 squid 我们是以 nobody 与 nogroup 来启动的！而 Process 与权限的关系是相当相当的重要的，因此，您必需要将上面刚刚建立的 cache_dir 更动整个目录的拥有者才行！如果是锻炼功力而已，那么随意建立一个目录更动一下他的拥有者与群组，就行了，不然的话，请依照您的硬盘规划好好的设计一番！

- 关于 cache_dir 重新建置的步骤：在上面的第 7 个步骤当中，我们必需以 squid -z 来重建一下 cache_dir 的格式，这个步骤在第一次启动 squid 时才做，其它时候就不需要进行了！而进行这个步骤之前，请务必将上面提到的两点注意事项先搞定，否则您的 squid 很难启动喔！

- 实际浏览器的查验：虽然上面第 8 个步骤已经确认了 squid 启动了，但是还不能肯定工作正常，这个时候如果您有 client 端的计算机，假设是 Windows 的 IE 好了，那么就赶紧来测试看看能不能使用 squid 啰！启动 IE 后，按下：

『工具』->
『Internet 选项』->
『联机』->
『局域网络设定』中，点选
『使用 Proxy 服务器』

并在网址列输入你的主机名称 (或者是 IP 均可)，然后按确定离开！然后在 IE 中按按看网页的设定有没有成功，如果可以读到网页的话，表示 squid 可以正常的被使用了！

- cache_dir 这个参数的意义与存取格式的类型：
  这个设定项目就是限制暂存区大小的地方啦，格式为：

```
cache_dir ufs /usr/local/squid/var/cache 100 16 256
```

上面的说明是：暂存区目录为 /usr/local/squid/var/cache ，而暂存空间大小为 100M ， 在这个暂存目录下有 16 个目录，而每个目录中又有 256 个目录 (你可以实际进入 /usr/local/squid/cache 当中去看看) 。如果你要改变暂存盘目录及这个目录的大小时，可以在这里修改！不建议修改 16, 256 这两个数值。另外，通常，如果是我们一般小型的区网 (不超过 10 个人) ，那设定个 500 MB 作为 cache 应该够了，如果你的硬盘够大，设定成 1000 MB 以上更好，当然，与第一个注意事项相符的，需要与您的实际硬盘大小以及 Partition 放置的目录互相配合才行。修改过这个指令后，要再重新启动 squid 之前，请先使用下面的指令来使你的目录可以进行存取的动作，否则你的 squid 是不会工作的！例如你将上面的参数修改成：

```
cache_dir ufs /usr/local/squid/var/cache 1000 16 256
```

然后再进行下面的指令：

```
rm -rf /usr/local/squid/var/cache
mkdir /usr/local/squid/var/cache
chown nobody:nogroup /usr/local/squid/var/cache
/usr/local/squid/sbin/squid -z
```

这样你的 squid 暂存目录就可以使用啰！另外，如果你在编译 (configure 过程中) 的时候有将 --enable-async-io 这一个参数加进来的话，将可以增加 aufs 这一个数据存取的格式！这个存取的格式可以将你的硬盘发挥到最极限的速度喔！虽然在 squid.conf 档案中有提及，这个 type 可能会有 bug 存在，不过，据鸟哥的使用结果，发现，没啥大问题！好用的很！所以，你可以将上面的咚咚改成下面的样子：

cache_dir aufs /usr/local/squid/var/cache 1000 16 256

这样就算已经完成了一个『很阳春』的小型 proxy 了！为什么说很阳春呢？这是因为这个 Proxy 并没有上层 Proxy 喂数据，所以 Client 端的任何要求这个 Proxy 都需要『自己去捉』啊！哇！那么这个 Proxy 还真累啊！没错啊，所以底下我们要再来谈一谈其它几个增加 proxy 效能的方法，好让大家的 Proxy Server 可以真的加快您浏览的速度啊！

内存与磁盘快取留存百分比设定

内存与磁盘快取在 squid.conf 当中的相关简易设定如下所示，至于更详细的说明则在下表之后进行解说：

```
# 与内存有关的设定：因为我的系统很小，所以只给 8 MB！如果您的物理内存
# 很大的情况下，例如 512 MB，可以考虑加大到 64 或 128 MB。
cache_mem                      8 MB
# 与磁盘容量有关的设定(注：下列的 90 与 95 是百分比 )
# 如果您的 cache_dir 所在磁盘很大时，可以考虑将 4096 改成 32768 KB
cache_swap_low                 90
cache_swap_high                95
maximum_object_size            4096 KB
# 与内存保存数据有关的设定
maximum_object_size_in_memory   8 KB
# 我们经由 dns 正反解以及 IP 的结果，记录在暂存区啊！
ipcache_size                   1024
ipcache_low                    90
ipcache_high                   95
fqdncache_size                 1024
```

内存的需求数量：
事实上，除了硬盘之外，内存可能是另一个相当重要的影响 Proxy 效能的因子！怎么说呢？因为 Proxy 会将数据存一份在 Cache 硬盘中，但是同时也会将数据暂存在内存当中啊，以加快未来使用者存取同一份数据的速度！所以啰，内存本来就会被 squid 的程序所消耗掉一些！一般来说，被 squid 消耗掉的内存约略每 1GBytes 的 cache_dir 空间就消耗 10MB 的内存容量，所以，如果以上面我们的设定为例 (cache1, cache2 共有 3GB)， 那么就有 30MB 以上的内存被消耗掉了！除此之外， squid 程序执行当中亦会额外的消耗掉一些物理内存，这部份占用掉的内存约为 10-20 MB 。

除了这些内存是必须要的之外，您还可以额外的指定一些内存来进行比较『热门』的数据存取！也就是说，可以额外的再加一些内存来帮助 squid 工作，而不仅只是上面提到的内存使用量！那个就是 cache_mem 这个设定参数的用途啦！所以，请特别留意啊，『 cache_mem 并不是指我要使用多少内存给 squid 使用，而是指 "我还要额外提供多少内存给 squid 使用" 的意思』！因此，假设我有 X GB 的磁盘快取空间，而且 squid 程序使用掉 15 MB 的内存，那么我 squid 使

用掉的内存就有：

X * 10 + 15 + "cache_mem 设定值"

您可以自行计算一下您的 squid 消耗掉多少内存喔！此外， squid 官方网站建议您的物理内存(不含 swap 的内存容量) 最好是上面数值的两倍，也就是说，假如我的快取容量为 3 GB ，cache_mem 设定为 16MB ，那么我的 squid 至少会消耗掉 3*10+15+16 = 61MB，则我的物理内存最好至少要有 122 MB 以上，才会有比较好的效能！当然，这个单指 Proxy 部分而已，如果您的该部主机还有负责其它的工作，呵呵！那么内存就得在累加上去啦！一般来说，如果您的 Proxy 很多人使用时，这个值越大越好，但是最好也要符合上面的需求喔！

关于磁盘容量的设定：
请注意，如果您的磁盘快取空间额满了，那么您的 squid 也就『挂点』了！因此，请随时注意你的磁盘快取空间！但是，要我天天去注意 squid 的 cache_dir 里面的容量，也太劳神了吧？！这个时候就必须要『cache_swap_low, cache_swap_high 』这些设定的帮忙了！如果以上面表格的设定为例时，他的说明是『当我的快取目录所占容量为总快取量的 95% 时，那么我的 squid 将会自动的将快取目录内的容量减低至剩下 90% 的容量！』注意，那个 90 与 95 为百分比例喔！以我们的设定为例，我的 cache_dir 总共有 3GB ，那么当快取空间被使用了 3*0.95=2.85GB 时，我的 squid 会自动的将 2.85GB 里面较旧的数据删除，使快取目录内剩下 3*0.9=2.7GB 的空间！这样能了解了吧？请特别注意，如果您的 proxy 容易很多人同时上线时，请将这两个数值更动一下，例如变更为 70 85 ，为什么呢？万一您的使用者突然间都上线了，然后下载大量的档案，那么『瞬间』可能会使您的 cache 目录超过额度，导致 cache 死掉！所以降低 cache_swap_high 对于大型 proxy 是有需要的！

是否需要记录捉到的数据：
还有一个小问题要说明的，是否使用者要求过的数据我都要记录呢？呵呵！那当然不是啦！如果都记录的话，万一像最近好多 Linux distribution 都释出了他们新版的 Linux ISO 档案，那些档案一个都有 600 MB 以上的容量，万一使用者下载个几个这样的档案，我们的 Proxy 想不爆掉都很难～所以这个时候就要限制一下需要记录的『最大档案』啦！在预设的情况下，我们的 squid 对于超过 4MB 的档案是不记录的(就是不放入 cache 的意思)，但是如果您的硬盘够大的话，我都喜欢将这个数值调大一点，为什么呢？万一我的使用者常常下载一些 10~20 MB 的套件档案，难道我每次都要到官方网站去下载一次吗？当然我不想这样啊！所以我通常将这个数值调大到 32MB ，或者是 32768 KB 啰！除了磁盘快取之外，内存的快取可以记录的最大档案容量也可以修正一下喔！但是我们的内存可就珍贵的多了！不要开太大，大约默认值就很不错了！

关于 IP 与完整主机名称的纪录：
如果能将 IP 与主机名称记录下来的话，搜寻的脚步也会加快一些的！所以以上面的表格为例，我们分别开启了 ipcache_size 与 fqdncache_size(后面接的数值是 bytes)的记忆容量，由于这些数据都是 ASCII 的数据，用不了什么空间，因此 1024 bytes 已经足够了！至于后面的 low 与 high 则与前面说明的相同！

---

acl 的用法与用途

在 squid.conf 这个档案里面最常看到的大概就是 acl 这个设定项目了！他可是整个 squid.conf 的重头戏啊！很多的来源与目的管制都是靠他来设定完成的呢！我们底下就稍微谈一谈这个重要的咚咚吧！首先，他的语法为：

```
<acl> <acl 名称> <acl 类型> <设定的内容>
```

上面那个 <acl 名称> 可以想成是一个昵称就是了！别想太多～就只是一个代名词，至于 acl 类型可就有趣的多了，他主要有下面这几大类：

以来源端来控制：

- src ip-address/netmask：主要控制『来源的 IP 地址』，例如『acl nckuip src 140.116.0.0/16』，这表示未来在 squid.conf 里面，任何使用到 nckuip 这个代名词时，就表示他是『来源为 140.116.0.0/16 的地址』！
- src addr1-addr2/netmask：主要控制『一段范围来源的 IP 地址』，例如『acl nckuevoffice src 140.116.44.120-140.116.44.130/24』就表示 nckuevoffice 这个代名词为来自 140.116.44.120 到 140.116.44.130 之间这 11 个 IP 的要求！』请注意，是『来源』喔！
- srcdomain .foo.com：主要控制『来源为一某个网域的计算机』的意思，例如：『acl vbirdhome srcdomain .vbird.org』，与 src 很类似，都是控制来源的客户端，只不过 src 控制的是 IP 而 srcdomain 则是控制 domain name 就是了！

以目的端来控制：

- dst ip-address/netmask：主要控制『目的端的 IP 地址』，与 src 类似，只不过是用来控制『目的端』的地址！
- dstdomain .foo.com：这个就没问题了吧？就是用来控制『目的端的网域』啰！与 srcdomain 类似喔！

以正规表示法的方式来控制：

- url_regex [-i] ^http:// ：除了上面两种基本的方法之外，我们也可以使用正规表示法的方式来控制『网域』的设定值呢！例如『 acl urlname url_regex ^http://linux\.vbird\.org.* 』这表示 urlname 代表的就是来自 http://linux.vbird.org 这个网站的『任何资料』，因为『 .* 』代表任意字符的意思啊！如果仅只是一些档名，例如 gif 这一类的档名时，要怎么作呢？就如底下的说明啦！
- urlpath_regex [-i] \.gif$：上面提到的是关于整个网址的名称，这里则是只要『URL 部分相同』就可以啦！例如『acl gifname urlpath_regex \.gif$ 』则是代表 gifname 代表的是 url 后面是 .gif 的网址，呵呵！那就是 gif 的图档附档名嘛！！这样应该不难理解了吧！

由于在 squid 当中很多时候都会用到一些 IP 网域啦，以及 domain name 等等的数据，这个时候这个 acl 就可以看做与 bash 里面的『变量』很类似啰！您可以参考看看的啦！只是他的用途相当的广泛，这个在底下我们会进一步的使用 acl 搭配各种设定值来进行说明的！

上层 Proxy 的选择与负载分流的设定方法

上层 Proxy 的设定方式： cache_peer
我们在上面的原理部分说明过，一部有效能的 Proxy server 必须要能利用上层 Proxy 所提供的效能，才能达到我们所想要的『加速』功能！那么当然大前提之下就是需要先找到『最佳效能的上层 Proxy 主机』啰！没错！就是这样的啦！在这里，我们以成功大学的学术网络为架构进行说明，如果您的网络架构并非为学术单位的话，请依照您能找到的上层 Proxy 来进行设定喔！需要特别留意的是，由于成大有许多的上层代理服务器，这些服务器都可以做为我个人 proxy 的 parent proxy 主机，不过，由于各个主要的 parent proxy 所提供的网域服务都不尽相同，所以我们得先将主要的几个服务器对应的网域给他找出来，底下是一个例子：

gate.ncku.edu.tw 主要服务 .com 的网域
gate3.ncku.edu.tw 主要服务 .net .edu 的网域
gate2.ncku.edu.tw 主要服务非 .com .net 与 .edu 的网域
proxy.ncku.edu.tw 主要服务任何网址

至于在 squid.conf 当中设定的参数则是『cache_peer 』这个项目，设定的样式为：

```
<cache_peer> <主机名称> <类别> <http_port> <icp_port> <其它参数>
类别：主要有上层(parent) 与同一层 (sibling) 两种，我们这里主要介绍的是
      上层 Proxy 也就是 parent 这一大类，如果您想要架设一个小型的 Proxy
      Cluster 的话，可以考虑组成 sibling 的功能，由于我们仅想要架设单部
      Proxy Server ，所以这里我们就不探讨 sibling 了！
http_port icp_port：就是我们最前面设定的啦！ Internet 上面预设是
      3128 3130 这两个！
其它参数：其它参数的部分就很重要了，主要有底下几个重要参数：
    proxy-only ：只取出上层 Proxy 的 cache 给 client ，并不会将上层 Proxy
        的数据存在自己的 cache 硬盘中；
    weight=n  ：权重的意思，因为我们可以指定多部上层 Proxy 主机，哪一部
        最重要？就可以利用这个 weight 来设定， n 越大表示这部 Proxy
        越重要！
    no-query  ：一般来说，如果要向 sibling 要求数据时，会向 sibling 送出
        icp 的要求封包，使用 no-query 就可以取消。一般来说，如果向上层
        Proxy 要求资料时，可以不需要发送 icp 封包，以降低主机的负担
    default    ：表示该部主机为预设的 Proxy 主机的意思；
    no-netdb-exchange：表示不向附近的 Proxy 主机送出 imcp 的封包要求
    no-digest  ：表示不向附近主机要求建立 digest 纪录表格。
范例：
```

```
cache_peer gate.ncku.edu.tw parent 3128 3130 no-digest no-netdb-exchange
cache_peer gate2.ncku.edu.tw parent 3128 3130 no-digest no-netdb-exchange
cache_peer gate3.ncku.edu.tw parent 3128 3130 no-digest no-netdb-exchange
cache_peer proxy.ncku.edu.tw parent 3128 3130 no-digest no-netdb-exchange
```

基本上，由于我们向上层 Proxy 要求数据的时候，也是需要时间的，因此，这里我们建议您不要设定太多的上层 proxy 主机，一般来说， 2~4 部也就足够了，太多部 上层Proxy 的情况反而可能会拖垮您的 Proxy 速度呢！OK！既然有了上层 Proxy ，而且每一部所负责的网域并不相同，所以我们当然就需要将网域分门别类啰！如何将这些网域分门别类呢？那就需要 acl 的帮忙啦！先提醒一下，由于我们的 Proxy 旨在帮助 client 端取得数据，所以使用 acl 的时候，是针对『目标』来进行设定的，因此，使用的是 dst 与 dstdomain 喔！在这里，我们将主要的网域分成底下这几大类：

```
0. 自己想要直接使用自己的 Proxy 捉取的数据
acl directip dst 140.116.44.0/24
acl directdn dstdomain .vbird.org tw.yaoo.com tw.news.yahoo.com

1. 成大附近的网域
acl nckudn dstdomain .ncku.edu.tw
acl nckuip dst 140.116.0.0/16 163.28.112.0/24 163.28.113.0/24 163.28.114.0/24 163.28.115.0/24
163.28.116.0/24 163.28.117.0/24

2. 台湾的网域 (请注意：底下为同一行)
acl twdn
dstdomain .tw .twnic.net .hinet.net .acer.net .wownet.net .seeder.net .silkera.net .neto.net
timenet.net tw.aunet.net .adsldns.org

3. 台湾的 IP (注意：底下为同一行)
acl twip dst 163.28.0.0/16 140.96.0.0/11 140.128.0.0/12 140.92.0.0/16 139.175.0.0/16
139.223.0.0/16 163.12.0.0/14 163.16.0.0/14 168.95.0.0/16 192.72.0.0/16 192.83.160.0/19
192.83.192.0/22 192.192.0.0/16 202.39.0.0/16 202.132.128.0/17 202.145.224.0/19 203.64.0.0/12
210.64.0.0/13 210.60.0.0/14

4. 一些商业、网域、教育、以及其它的网域
acl comdn dstdomain .com
acl netdn dstdomain .net
acl edudn dstdomain .edu
```

这样子就作好了基础的工作了！接着下来，我们要开始来设定『各个主要的网域要透过哪一个上层 Proxy 来进行数据的取得？』这里就有点难度啦！得小心在意一下！我们的基本假设为这样：

gate.ncku.edu.tw 仅负责 .com 的网域；
gate3.ncku.edu.tw 仅负责 .net 与 .edu 的网域
gate2.ncku.edu.tw 不负责上面的三个网域

proxy.ncku.edu.tw 不负责上面三个网域
这么一来，使用 cache_peer_access 这个设定项目时，我们的设定会变成这样喔！

```
# 主要的格式范例：
# <cache_peer_access> <上层 Proxy > <allow|deny> <acl 名称>
cache_peer_access gate.ncku.edu.tw  allow comdn
cache_peer_access gate.ncku.edu.tw  deny  !comdn
cache_peer_access gate3.ncku.edu.tw allow netdn
cache_peer_access gate3.ncku.edu.tw allow edudn
cache_peer_access gate3.ncku.edu.tw deny  !netdn
cache_peer_access gate3.ncku.edu.tw deny  !edudn
cache_peer_access gate2.ncku.edu.tw deny  comdn
cache_peer_access gate2.ncku.edu.tw deny  netdn
cache_peer_access gate2.ncku.edu.tw deny  edudn
cache_peer_access gate2.ncku.edu.tw deny  directdn
cache_peer_access gate2.ncku.edu.tw deny  directip
cache_peer_access gate2.ncku.edu.tw deny  twdn
cache_peer_access gate2.ncku.edu.tw deny  twip
cache_peer_access gate2.ncku.edu.tw deny  nckudn
cache_peer_access gate2.ncku.edu.tw deny  nckuip
cache_peer_access proxy.ncku.edu.tw deny  comdn
cache_peer_access proxy.ncku.edu.tw deny  netdn
cache_peer_access proxy.ncku.edu.tw deny  edudn
cache_peer_access proxy.ncku.edu.tw deny  directdn
cache_peer_access proxy.ncku.edu.tw deny  directip
cache_peer_access proxy.ncku.edu.tw deny  twdn
cache_peer_access proxy.ncku.edu.tw deny  twip
cache_peer_access proxy.ncku.edu.tw deny  nckudn
cache_peer_access proxy.ncku.edu.tw deny  nckuip
```

经过上面这个动作，我们就可以将 .com 的要求经过 gate.ncku.edu.tw 这一部，并且将 .net 与 .edu 给 gate3.ncku.edu.tw 来服务！此外，将没有被自己定义出来的 IP 或者是网域就丢给 gate2 以及 proxy 这两部来服务，至于我们既然是在台湾，所以自己定义出来的四个 acl 名字 (directdn, directip, twdn, twip) 当然是直接交给我们自己的 Proxy 来捉取了，相信速度上面应该不会有任何的影响的！因此，那四个主要的 acl 名称将在后面继续介绍如何不要透过上层 Proxy 来捉取数据喔！ (请参考 always_direct 与 never_direct 的设定值)

不要进行 cache 的设定值：
就如同我们在原理的部分提到的，有些网站的数据事实上并没有做为 cache 的需要，例如 CGI 档案就是一个很鲜明的案例！所以，一般而言， squid 在一开始就会自动帮我们设定好底下这三行：

```
hierarchy_stoplist      cgi-bin ?
acl QUERY urlpath_regex cgi-bin \?
```

```
no_cache deny QUERY
```

上面这三行的意思是：『只要在网址列出现了 cgi-bin 的字样时，该网页数据就不进行 cache 的动作！』这样一来，可以避免我们的主机存放太多的垃圾啊！ ^_^！当然啦，在某些时候，例如您所在的网域真的是很慢，同时又没有适合的上层 Proxy 的环境下，那么将 cgi 之类的网页 cache 下来，也未尝不是一个可以节省浏览时间的一个方法！如果确定要将 cgi-bin 底下的网页也存下来的话，那么就将上面三行批注掉啰！

与时间相关的设定值 (connect_timeout, request_timeout)

由于 Proxy 的设定除了防火墙的额外功能外，最主要的目的还是在于将网页数据 cache 下来！既然如此的话，那么与 Server 的联机时间的确认就显的很重要啦！怎么说呢？万一今天您有设定三部上层 Proxy ，每一部的要求时间都控制在 5 分钟，如此一来，呵呵！万一今天刚好三部上层 Proxy 都挂点，难道我们就宿命似的要等待 15 分钟吗？当然不是啦！所以这个时候，我们可以将联机的等待时间缩短，好让 Proxy 可以发挥更为强大的功能啊！

```
# 1. 关于 cache 的更新时间趋势
# 我们在快取当中的数据总是需要更新的对吧！那么如何设定更新的时间呢？
# <refresh_pattern> <regex> <最小时间> <百分比> <最大时间>
# 注意，那个 regex 指的是『目标』，这个『目标』大部分为网址，而这个网址
# 使用正规表示法来表示就是了！上面的意义是说：最小的时间内 (分钟)，如果
# 档案有更动，就直接更新，或者是目标档案上次更新的时间到现在已经经过最大
# 时间的『百分比』时，就予以更新！范例如下：
refresh_pattern ^ftp:           1440    20%     10080
refresh_pattern ^gopher:        1440    0%      1440
refresh_pattern .               0       20%     4320
# 第一行的意思是，如果网址裂开头是 ftp 的话，那么在一天(1440 分钟)后，如果
# proxy 再次取用这个档案时，则 cache 内的数据会被更新！

# 2. 关于联机时间限制
# connect_timeout 指的是连接到其它主机的时间多久之后算失败，预设是两分钟
#                 不过我觉得我的机器联机速度还算快速，所以减低了时间
# peer_connect_timeout 指的是连接到上层 Proxy 多久不成功就算失败，
#                 由于上层 Proxy 与我的环境息息相关，不可能连不上，
#                 所以我这里将时间调整的很短！
# request_timeout 指的是，联机上了，但是要求的时间会多长？
# persistent_request_timeout 指的是连续要求时间会有多长？
connect_timeout 30 seconds
peer_connect_timeout 10 seconds
request_timeout 1 minutes
persistent_request_timeout 20 seconds
# 请特别注意喔！上面的设定当中，是因为我的环境『还算不错』，所以我将
```

```
# 时间调整的很短，因为我晓得我的环境当中不会有花很长时间的状况，
# 这里请依照您的网络环境来调整喔！或者干脆不要设定也没有关系！
```

总是系统自己来捉数据(always_direct)

刚刚我们在主机分流的地方有提过，有些数据我们要让我们自己的 Proxy 去捉取数据即可，而不透过上层 Proxy ，那么怎么达到这样的目的呢？也不是很难啦！就告诉 Proxy 不要去捉就好啦！怎么告诉他呢？使用 always_direct 或 never_direct 即可！

```
# 1. 使用 acl 先定义出要直接或者不要直接去的网址或 IP ，
#    这个部分我们刚刚在上面已经设定好了，就是 directip 与 directdn，
#    还有 twdn 与 twip 这几个咚咚！
# 2. 如果是直接要 Proxy 出去捉资料，可以使用 always_direct
always_direct allow directip directdn

# 3. 如果一定『拒绝经由上层 Proxy 出去』的话，可以使用 never_direct
never_direct deny twdn twip

# never_direct allow 表示一定会经由上层 Proxy 来捉资料，
# never_direct deny  当然就表示一定是自己向外头捉资料啰！
```

事实上，这个东西可以让我们的 Proxy 变的很灵活！假设这样的一个案例，我们自己有一个内部的 WWW 网站，这个网站的网址为 192.168.0.100 ，如果我要经由上层 Proxy 去捉数据的话，那不就完蛋了～因为这个是『私有 IP 』的网域啊！所以，我将他写入 directip 那个 acl 的设定当中，如此一来，呵呵！我们的 Proxy 会自动的经由自己的 route table 去到内部网域读取数据给你，您根本不需要变更您的其它设定就可以自由自在的读取内部与外部的主机数据！此外，如果您发现同一网域还有其它的 WWW 主机，把这些主机的 IP 或主机名称写入 directdn 或 directip 的 acl 设定当中吧！因为在同一网域时，您自己去捉一定会比上层 proxy 捉完之后再传给你来的快吧！

当然还不只如此啦，有的 WWW 主机由于设定的关系，他们并不允许我们的上层 Proxy 来捉取数据，最常见的例子就是类似总图对校内的 client 端开放的图书查寻的软件了！因为如果开放了这些上层 proxy 的话，那么全台湾所有的人只要将他们的浏览器 proxy 设定为成大的上层 proxy 主机，就可以使用成大的资源了！那岂不麻烦？因为这些资源是需要花费经费的啊！这个时候，您也就必须要让这些网址经由我们的 Proxy 自己去捉取啰！这样可以了解乎！

限制使用 proxy 使用者与 proxy 目标的方式 (acl and http_access)

既然 proxy 有一定的风险存在，自然就不能让任何人都能使用你的 proxy 主机啰！没错！所以我们必须要管制联机的使用者！管制的方法真是简单的很！就是使用 acl 配合 http_access 即可！在预设的情况中， squid 已经帮我们设定好一些安全的可以联机的 port 了，此外，也只有本机可以使用 proxy 功能呢！

那么万一如果我样让内部 192.168.0.0/24 这个网域的使用者可以使用我的 proxy 呢？该如何设定？呵呵！您可以这样做：

```
# 1. 先设定这个内部网域的 acl 名称
acl inside src 192.168.0.0/24

# 2. 设定 http_access 让他可以使用
http_access allow inside
http_access deny  all
```

那个 http_access deny all 是系统预设的项目！刚刚我们在 最简单的 squid.conf 设定 时已经将他改成 http_access allow all 了！所以请记得将他给改回来啊！不然的话，您的 Proxy 很有可能会被人家利用喔！

这里再提供一个值得思考的咚咚，如果您跟我一样，都是使用拨接的 ADSL ，这样一来，由于我们的 IP 都不是固定的，如果要让我们的 ADSL 拨接的 client 可以使用我们刚刚设定的 Proxy 时，该怎么办？啊！这样就不能使用『 acl 配合 src 』的设定方式了吗？呵呵！当然不是，您可以这样想象：

25. 我先申请一个动态 DNS 的网域名称，例如我的 tsai.adsldns.org ；
26. 虽然我可以直接在 squid.conf 当中设定 acl 并使用 srcdomain 来设定我的 tsai.adsldns.org ，但是很抱歉的是，如果我的 tsai.adsldns.org 来连接到 Proxy 主机时，事实上，我的拨接制 IP 反查得到的主机名称一定不是 tsai.adsldns.org ，如此一来则 srcdomain 一点用处也没有了；
27. 再换个方式，如果我写一支 script 来侦测 tsai.adsldns.org 所对应的 IP 呢？并且将他写入一个自订的设定档当中，这样一来，这个档案会随时记录最新的 tsai.adsldns.org 的 IP ，如此一来，我就可以使用 acl 配合 src 的设定方式了！

很麻烦吗？一点也不会，整个 script 可以像底下这样：

```
[root@test root]# cd /usr/local/squid/etc
[root@test etc]# vi squid.allow.sh
#!/bin/bash
# 这支程序可以用来查寻您的 IP 喔!

# 1. 请输入您的主机名称，请注意，如果有两个以上的主机名称，
#     请分别以空格分开各个主机名称
hostnames='tsai.adsldns.org test.adsldns.org'
basedir=/usr/local/squid/etc/
email=root@localhost
```

```
squid=/usr/local/squid/sbin/squid

# 2. 以下为程序段，看看就好了！
[ -f $basedir/squid.allow.hosts.raw ] || \
        touch $basedir/squid.allow.hosts.raw

cat /dev/null > $basedir/squid.allow.hosts.now
runornot=no

for host in $hostnames
do
        hostip=`host $host | awk '{print $4}'`
        if [ "$hostip" == "out;" ]; then
                echo 'Proxy 回应：没有 DNS 的讯息！ ' |\
                mail -s 'Proxy 主机响应' $email
                exit
        fi
        fileraw="$basedir"/squid.allow.`echo $host|cut -d '.' -f1`
        [ -f $fileraw ] || touch $fileraw
        hostraw=`cat $fileraw`
        if [ "$hostraw" != "$hostip" ]; then
                runornot="yes"
                echo $hostip > $fileraw
        fi
        echo $hostip >> $basedir/squid.allow.hosts.now
done

if [ "$runornot" == "yes" ]; then
        cat $basedir/squid.allow.hosts.raw >  $basedir/squid.allow.hosts
        cat $basedir/squid.allow.hosts.now >> $basedir/squid.allow.hosts
        $squid -k reconfigure
        mail -s 'Proxy 主机响应！Client IP 已经改变' $email < $basedir/squid.allow.hosts
fi
```

如此一来，我将可以把我的随时最新的 IP 纪录在 /usr/local/squid/etc/squid.allow.hosts
这个档案当中！那么要如何更新这个档案的内容在 squid 的设定档中呢！就这样设定即可：

```
# 1. 先设定这个档案的名称吧！
acl allowhost src "/usr/local/squid/etc/squid.allow.hosts"

# 2. 设定 http_access 让他可以使用
http_access allow allowhost
http_access deny  all
```

然后将上面这个 squid.allow.sh 给他丢进去 crontab 当中，我预设都是在 30 分钟跑一次！

```
[root @test root]# vi /etc/crontab
10,30 * * * * root /usr/local/squid/etc/squid.allow.sh
```

这样就完整的啦！ ^_^

额外的功能参数

除了上面提到的这些关于网页快取的功能之外， Proxy 还可以帮我们进行 FTP 的服务取得资料喔！我们可以透过浏览器，经由 proxy 提供的 FTP 功能来登入对方主机，当然，对方主机必须要能够提供匿名登入啊！好了，我们来看看要怎样设定呢？

```
# 与 FTP 有关的设定项目，主要是针对被动式联机方式来设定喔！
ftp_user Squid@
ftp_passive on

# 主要与 DNS 的设定值有关，如果在高负载的 Proxy 环境下，可以考虑将
# dns_children 提高到 20 左右，这个值最大为 32
dns_timeout 1 minutes
hosts_file /etc/hosts
```

还有一些额外的范例可以参考看看喔：
鸟哥的范例：http://linux.vbird.org/linux_server/0420squid/0420squid_vbird_ex
成大 gate.ncku.edu.tw 的 squid.conf 设定：
http://turtle.ee.ncku.edu.tw/~tung/proxy/squid.conf.ncku
电机系 turtle.ee.ncku.edu.tw 的 squid.conf 设定：
http://turtle.ee.ncku.edu.tw/~tung/proxy/squid.conf.turtle
台南学校范例 proxy.school.tn.edu.tw 的 squid.conf 设定：
http://turtle.ee.ncku.edu.tw/~tung/proxy/squid.conf.school.tn

---

Client 端设定

既然 proxy 是给浏览器用的，那么自然在浏览器上面就需要设定一些参数啰！呵呵，没错！那么如何设定呢？由于不同的浏览器在设定 Proxy 的地方也都不同，所以底下我们介绍目前比较常见的两款浏览器，分别是 Netscape 以及 IE 的设定，至于其它的浏览器，请参考各浏览器的相关说明啊！

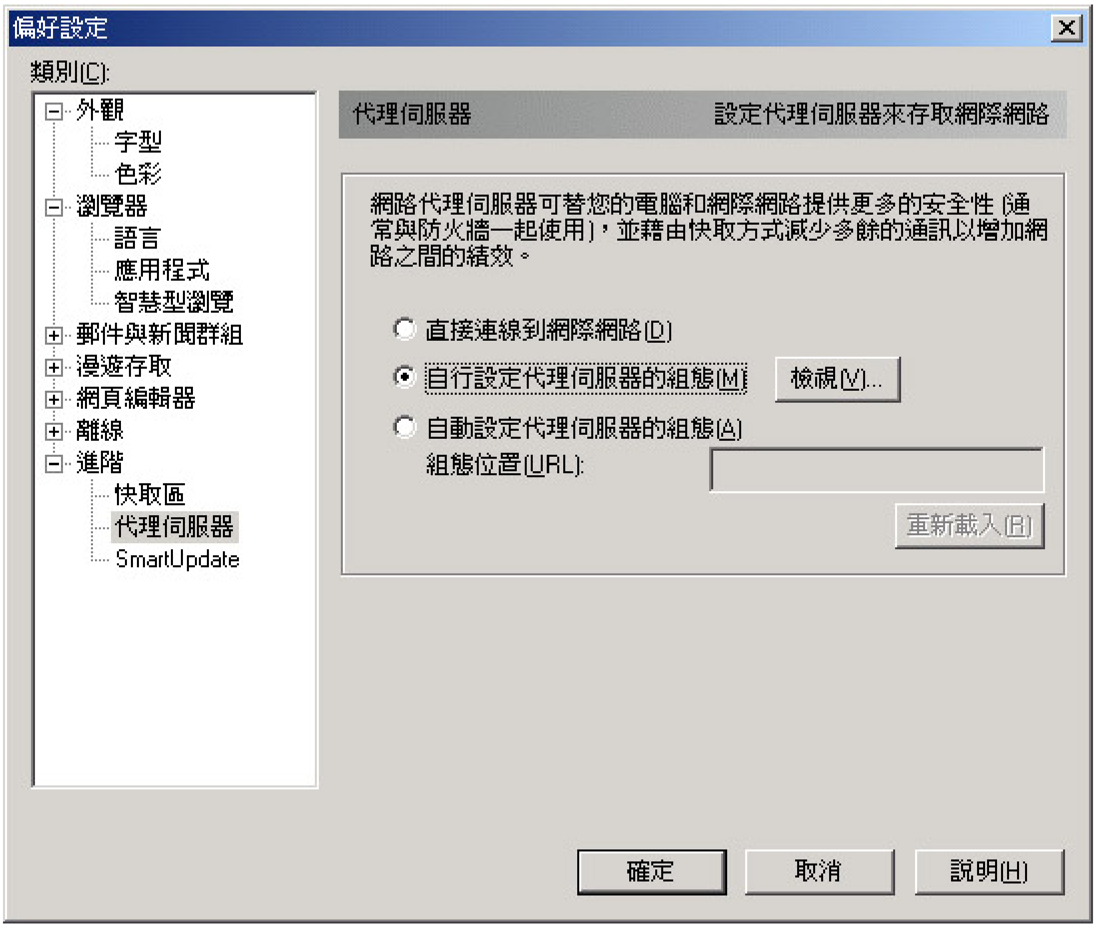
Netscape

Netscape 的设定并不难，只要修改一个小地方即可！

1. 开启 Netscape 之后，启用『编辑』并选择『偏好设定』项目，如下所示：

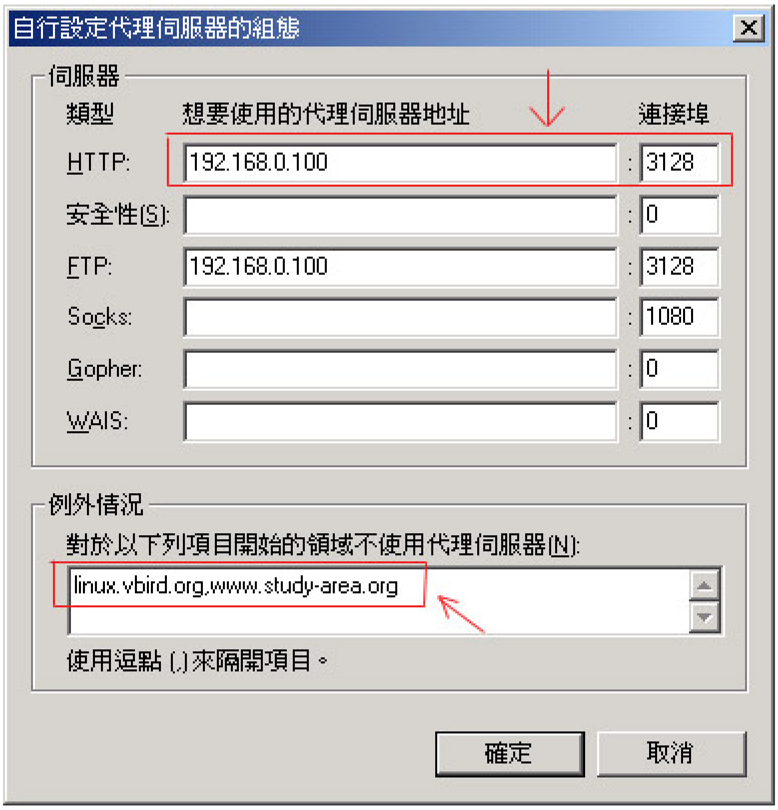


2. 在出现的画面当中，先按下『进阶』左边的展开模式，然后选择底下的『代理服务器』，则画面会出现如下的模样。之后，在右边的窗口当中选择『自行设定代理服务器的组态』项目，并且再按下『检视』，则到下一步去设定：



3. 在出现的方框中，由于我们通常仅针对 WWW 进行快取，所以可以仅针对 HTTP 的项目进行设定！在下面的画面当中，请输入『Proxy 主机的 IP 或者是 domain name 』，以及

『连接的 port number』，这样就可以了。不过，如果有某些网域您想要直接让你的 PC 去捉取数据时，可以将该网域名称填写在最下方的方框中！以下面的画面为例，我的 PC 要连接到 linux.vbird.org 以及 www.study-area.org 时，就可以不透过 Proxy 啦！这个功能也挺适合可以自行调配流量、流速的朋友！



4. 再按下『确定』之后，立刻就生效了！

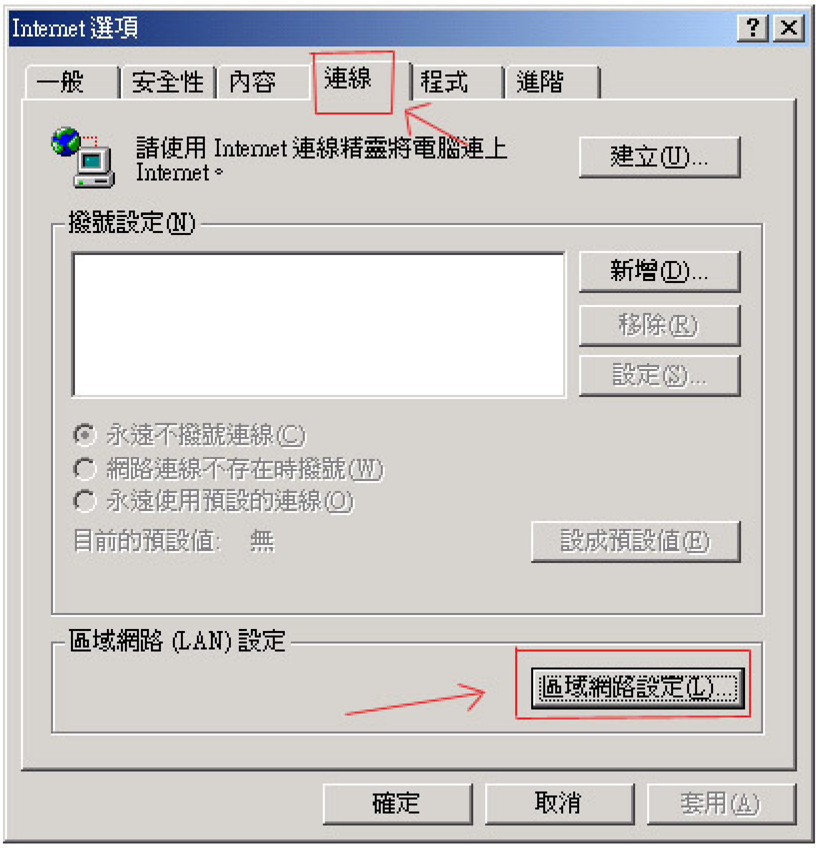
Internet Explorer

在 IE 的设定上面也一点都不难啊！

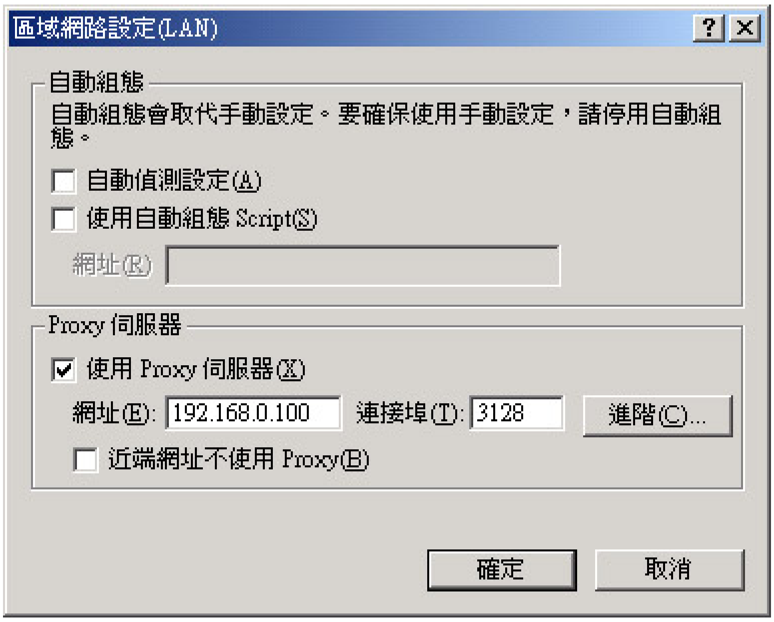
5. 开启 IE ，然后在『工具』内选择『Internet 选项』：



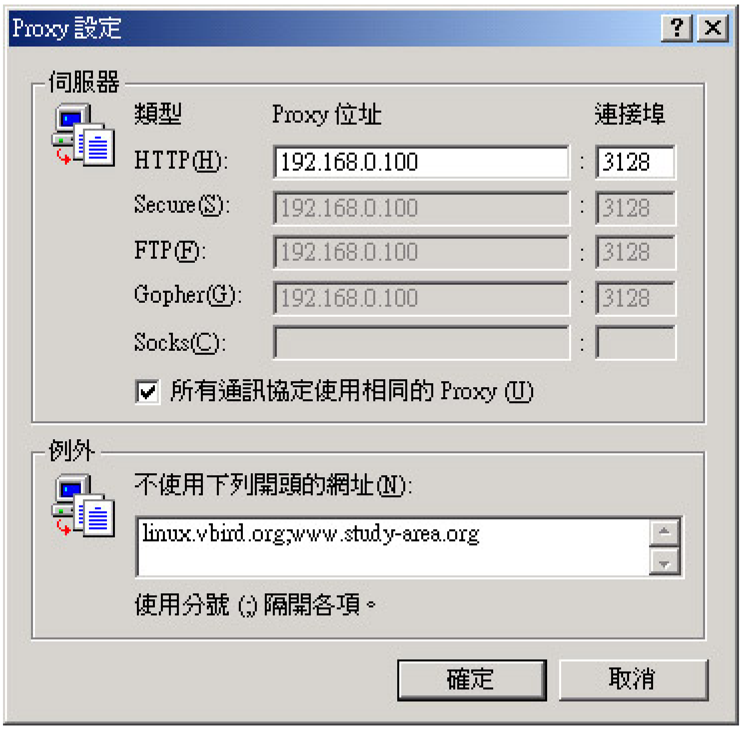
6. 在开启的窗口内选择『联机』并点选下方的『局域网络设定』：



7. 在出现的方框中，先在下方打勾起『使用 Proxy 服务器』，然后依序在网址列填入『 Proxy 的网址或者是 IP 』以及连接埠填入『 Proxy 所开放的 port number』，如下方的设定即可。



8. 另外，如果有某些网域不想透过这个上层 Proxy 的话，点选上图的 『进阶』那一项，在出现的框框的最底下，可以填入你不想透过 Proxy 捉取的网站网址，如下所示：



如此一来则设定好了 Proxy Server 啰！

Server 端进阶设定

既然一些主机的设定已经搞定了，接着下来又是到了 Server 的安全与进阶设定时间啦！那么在安全的设定方面，最重要的自然又是我们一再强调的登录档的分析啰！如何分析登录档呢？难道

又要自己动手写分析的 scripts ？？呵呵！不用麻烦了，我们可以使用前辈智慧的结晶，直接有软件可以来进行分析喔！还有，既然强调 Proxy 可能会被滥用，所以当然要适当的管理了！这个时候的防火墙设定又要出现啦！呵呵！赶紧来看一下怎么处理吧！

末端资料分析 pwebstat

事实上， squid 已经有众多的登录文件分析软件了，而且大多是免费的 (http://www.squid-cache.org/Scripts/) ，您可以依照自己的喜好来加以安装与分析你的 squid 喔！我这里仅介绍目前很常被使用的一套软件，也就是 pwebstats 这一套！你可以在 pwebstats 的官方网站上面查得更新的数据 (http://martin.gleeson.com/pwebstats/installation.html)。不过，由于 pwebstats 在安装的时候需要使用到其它的函式库，因此，你必须要先安装 fly 这套软件才行啊！而在安装 fly 之前，又需要先安装一些必备的图示用的函式库才行，那就是类似 gd, libpng, zlib 等等的套件喔！听起来似乎很多东西要做，但是事实上却非常简单的啦！好吧！那么我们就一步一步的来安装这个 pwebstats 吧！

安装所需要的套件数据：
除了系统自己可能已经安装好的 gd, libpng, zlib 之外，您可以由文末提供的官方网站的连结来取得最新的 fly 与 pwebstats 套件，而如果您不需要使用最新的资料，那么也可以经由 鸟哥的私房菜 提供的比较旧的套件版本来安装 (http://linux.vbird.org/download/)。我这里假设您已经将所需要的 fly-2.0.0.tar.gz 与 pwebstats-1.3.8.tar.gz 套件都下载到 /root 底下了，那么您可以这样做：

```
1. 确认一下到底有没有我们需要的一些相关套件？
[root@test root]# rpm -qa | egrep '(^gd-|^zlib-|^libpng-)'
libpng-devel-1.0.14-0.7x.4
gd-1.8.4-4
zlib-1.1.3-25.7
zlib-devel-1.1.3-25.7
libpng-1.0.14-0.7x.4
gd-devel-1.8.4-4
# 请注意，我是以 Red Hat 7.2 为例，如果您的系统并非 RH 7.2 ，那么可能
# 套件后面接的版本会不太一样，不过，至少都需要有上面的几样套件就是了！
# 就是 gd, zlib, libpng 这三个套件啦！有的 distribution 还会有 xxx-devel
# 之类的名称，那个也需要安装喔！如果发现没有安装的话，请拿出您的原版光盘片，
# 将他 mount 上之后，好好的将他搜寻一下，并且安装上去吧！很重要，一定要
# 安装这些套件之后，底下的动作才会成功呢！

2. 开始安装 fly 这支程序
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/fly-2.0.0.tar.gz
# ....会产生一个 fly-2.0.0 的目录
[root@test src]# cd fly-2.0.0
```

```
[root@test fly-2.0.0]# make
注：
# ....如果没有出现底下的字样那么就是成功了！
/usr/bin/ld: cannot find -lttf (或者是 ljpeg ...)
collect2: ld returned 1 exit status
make: *** [fly] Error 1
# 如果是出现 warning 的话，还不打紧，因为仅只是警告而已，所以问题不大，
# 但是如果出现上面的字样时，就会连带出现第三行的 Error ，那就是编译失败了
# 果真如此的话，请先寻找一下，以上面为例，找不到 lttf ，而 lttf 其实
# 就是 libttf 檔名啦！(所以 ljpeg 就是 libjpeg )所以给他下达：
loate libttf
/usr/lib/libttf.so.2
# 嗯！找到了！其实是 libttf.so.2 啦！如果你的系统跟我的不同，有可能档名
# 会不一样，所以请依照你的屏幕显示的讯息来下达指令喔！但是这个编译程序
# 有点笨，不认识这个档案，所以我们要欺骗他一下，就是下达：
ln -s /usr/lib/libttf.so.2 /usr/lib/libttf.so
# 这样就可以啦！如果还有找不到的档案，同样的方式给他连结一下即可！然后
# 再次的给他执行一次 make 即可！最后会产生一个档名为 fly 的执行档，
# 如果没有产生执行档，那就是有问题啦！请赶紧再回头去查一查是否有某些
# 套件没有安装，或者没有依照上面的方式建立连结档案！

[root@test fly-2.0.0]# cp fly /usr/local/bin
我们先将 fly 这支程序给他复制到 /usr/local/bin 这个目录去，可以直接使用
这样就 OK 了！

3. 开始安装 pwebstats 套件
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/pwebstats-1.3.8.tar.gz
# 此时会产生 /usr/local/src/pwebstats-1.3.8 这个目录，这目录里面
# 已经是可以执行的数据了，不需要进行任何的编译，所以我们将他直接移动到
# /usr/local/ 里面去！
[root@test src]# mv pwebstats-1.3.8 /usr/local/pwebstats
[root@test src]# cd /usr/local/pwebstats
[root@test pwebstats]# vi pwebstats <==这个是主要的执行档！
# 由于我们的 perl 这个程序语言的执行档是放置在 /usr/bin/perl ，但是
# 这支程序预设是书写为：
#!/usr/local/bin/perl
# 将上面改写为
#!/usr/bin/perl
# 这样就可以了！继续下一步！

4. 开始设定 squid 的输出输入参数：
前提要件：
a. 我的 WWW 主页目录在 /var/www/html
```

```
b. 我的 pwebstats 整个数据库放置在 /usr/local/pwebstats
c. 我的 squid 登录档放置在 /usr/local/squid/var/logs/access.log.0
d. 我的 squid 的执行档为 /usr/local/squid/sbin/squid
e. 我的 pwebstats 输出的数据放置在 /var/www/html/pwebstats
f. 我的 fly 是放在 /usr/local/bin/fly
[root@test pwebstats]# mkdir /var/www/html/pwebstats<=请依您的主机而定
[root@test pwebstats]# cd /usr/local/pwebstats/conf
[root@test conf]# vi squid-proxy.conf  <==这个就是主要的设定档!
# 底下请填写你的主机的『昵称』!
server:My_Proxy_Server
# 这个是显示在网页上面的标题内容!
Server_header:我的代理服务器
# squid 登录档放置的完整档名
logfile:/usr/local/squid/var/logs/access.log.0
# 我们使用的就是 squid 的登录档啊!所以这里不需改变
logtype:squid
# 刚刚提到的，网页数据要输出到底下的目录
outdir:/var/www/html/pwebstats
# 一些图像档案的预设目录，这里我们使用 pwebstats 提供的目录!
templates:/usr/local/pwebstats/templates
# 嗯!我们就天天来进行一次吧!可以用 weekly 或 daily
interval:daily
# 是否在执行的过程中给他输出讯息呢?好吧!
verbose:true
# fly 放置的目录，刚刚我们移动的目录啊!
fly_prog:/usr/local/bin/fly
# 本机端的要求，这里可以不用设定喔!
local_patt:
# 列出几部 client 计算机?我给他来个 50 部好了!
host_threshold:50
# 列出几部 Server (被要求资料者)，还是给个 100 部吧!
remote_host_threshold:100
# 统计的次数，给他 100 次好了!
item_threshold:100
# 做多少个 domain 的分析?给他 15 个就够多了!
domain_threshold:15
# 底下保留默认值即可! (最后面的 dns 反查可以启动!)
exclude_reqs:true
#complete_exclude_host:
#complete_exclude_url_patt:
#complete_exclude_user:
dns_lookup:true
# 事实上，只要上面粗体字的地方写对就好了!其它的保留默认值即可!
```

```
5. 测试执行一次看看：
[root@test conf]# /usr/local/squid/sbin/squid -k rotate
上面这个动作会将 squid 的登录档进行 logrotate 的动作喔！就会更新了
/usr/local/squid/var/logs/access.log.0 这个档案啦！
[root@test conf]# /usr/local/pwebstats/pwebstats -c  \
> /usr/local/pwebstats/conf/squid-proxy.conf
.....(略)......
-- Reading in log file /usr/local/squid/var/logs/access.log.0:
   The logfile has 27 entries.
   Processing...
   0%                    50%                      100%
   |-----------------------|------------------------|
   ##################################################
   Finished.
.....(略)......
呵呵！这样就是成功啦！！！！

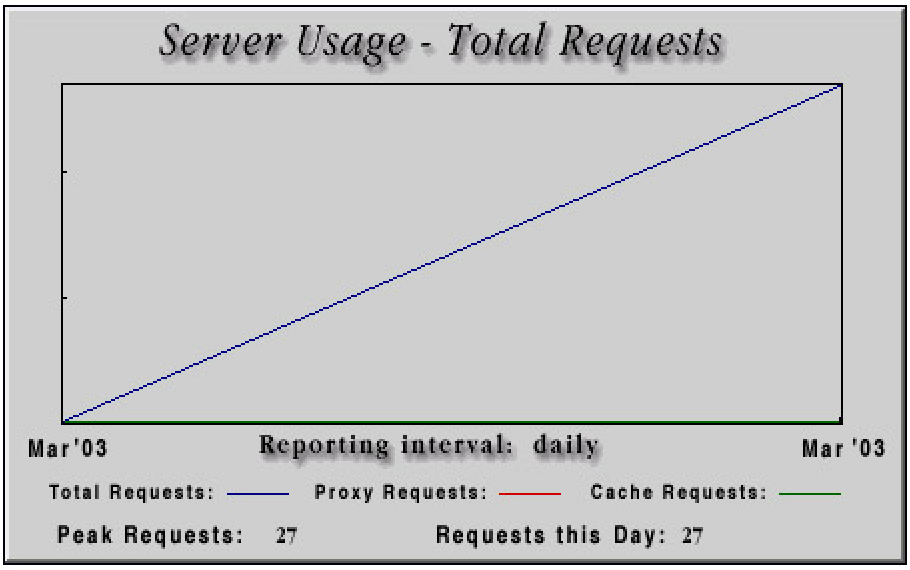
6. 将每天执行的指令写成 scripts 吧！
# 我将他写在/usr/local/pwebstats/pwebstats.sh 这个档案当中喔！
[root@test conf]# vi /usr/local/pwebstats/pwebstats.sh
#!/bin/bash
/usr/local/squid/sbin/squid -k rotate
sync; sleep 5s
/usr/local/pwebstats/pwebstats -c /usr/local/pwebstats/conf/squid-proxy.conf
[root@test conf]# chmod 755 /usr/local/pwebstats/pwebstats.sh
[root@test conf]# vi /etc/crontab
# 加入底下这一行吧！
59 2 * * * root /usr/local/pwebstats/pwebstats.sh >/dev/null 2>&1
```

很简单吧！这样就安装完毕了！不过，由于我们安装的 fly 可能有点问题啦！就我的系统来说，这个 fly 总是编的怪怪的～如果您的 fly 无法编译成功的话，那么直接由鸟哥的私房菜提供的档案来直些下载使用算了！ http://linux.vbird.org/download 找一下 fly 的 binary ，下载完毕之后，请记得：

```
gzip -d fly.gz
chmod 755 fly
mv fly /usr/local/bin
```

这样就可以了！而刚刚我们不是有执行过一次吗？你的登录数据会被放置在 /var/www/html/pwebstats 这个目录当中！假如我的首页就是 /var/www/html/ 的话，那么我的网址列输入 http://myIP/pwebstats 就可以看到类似下图：

我的代理伺服器



Day 1 (29 Mar 2003 - 29 Mar 2003) 27

These statistics produced by pwebstats.

在按下了『Day 1』之后，会出现如下的画面：

我的代理伺服器

Period covered by these statistics: *1 days* (29 Mar 2003 - 29 Mar 2003)

Total requests handled this day: **27**

Requests Satisfied by the cache:	0	
Requests Proxied	27	
Total number of requests served:	27	***
Bytes sent for cache requests:	0	
Bytes sent for proxy requests:	89,100	
Total number of bytes sent:	89,100	***

Cache Hit Rate	
Requests:	**0.00%**
Bytes:	**0.00%**

Number of Mb sent by this server:	**0.08**
Number of hosts using this server:	**1**

Average number of requests/day:	**27**
Average number of requests/hour:	**27.00**
Average number of requests/minute:	**0.45**

Hosts accessing this server, ordered by number of accesses:

呵呵！更详细的内容您就可以自行看看啰！加油的啦！！

末端资料分析 sarg

除了上面介绍的 pwebstats 之外，其实还有一套相当棒而且功能相当强悍的分析软件，那就是 Squid Analysis Report Generator (Squid 分析报告制作者)，他的官方网站在：http://web.onda.com.br/orso/sarg.html，他的原理相当的简单，就是将 logfile 拿出来，然后进行一下解析，依据不同的时间、网站、与热门网站等等来进行数据的输出，由于输出的结果实在是太详细了！所以.....呵呵！如果你是老板的话，用这个软件会让你『爱不释手』啊！因为每个人的每个小动作都会被记录下来，我的天呐！当我第一次看到这个分析的画面时，真的给他吓了老大一跳得说～因为连每个 IP 在『每个小时所连上的每个网站数据』都有纪录～～害怕了吧～

不过，有优点就有缺点啦！怎么说呢？因为 SARG 功能太强大了，所以记录的『数据量』就实在是多了点，如果您的 Proxy 网站属于那种很大流量的网站时，那么就不要使用『日报表』，也就是每天产生一份报表的那种方式！那么由于数据一天可能会有几 MB 的数据，一两个月还没有关系，如果记录了几年，那么光是这些记录就会花掉好几 GB 的硬盘空间了～此外，也可以使用『覆

盖旧有数据』的方式不要留存旧数据，这样也可以节省硬盘的空间啦！

这个 SARG 软件已经出到 1.4 版，更好的是，他支援『多国语系』呢！目前鸟哥已经翻译了一部分的资料，希望能对大家有点帮助啊！对于这个套件，你可以到官方网站上下载最新的版本 (http://web.onda.com.br/orso/sarg.html) ，而鸟哥这里也有提供啦!我提供的版本是 1.4 的 (在 2003/03/16 释出的版本)，已经经过了 patch (套件修补) 的动作，并且加入中文化的语言档案，所以直接下载就有中文显示啰！请到档案下载中心下载：

- o http://linux.vbird.org/download#squid_ap

整个安装与执行的过程很简单的！我们来试看看吧！(注：假设您已经将 sarg-1.4.taiwan_big5.tar.gz 放置在 /root 底下了！)：

```
# 1. 解压缩：
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/sarg-1.4.taiwan_big5.tar.gz
....(略).....
这个时候会产生一个名为 sarg-1.4 的目录出来!

# 2. 设定、编译与安装
[root@test src]# cd sarg-1.4
[root@test sarg-1.4]# mkdir /usr/local/sarg
[root@test sarg-1.4]# ./configure --prefix=/usr/local/sarg  \
> --enable-mandir=/usr/local/sarg/man1  \
> --enable-bindir=/usr/local/sarg/bin  \
> && make && make install
这样可以快速的设定、编译与安装，一次完成!
安装的数据当中：
a. 设定档 /usr/local/sarg/sarg.conf
b. 执行档 /usr/local/sarg/bin/sarg
c. 说明档 /usr/local/sarg/man1/sarg.1

# 3. 额外设定 ( 说明文件的路径 )
[root@test root]# vi /etc/man.config (有时为 man.conf 檔名不同！)
新增加这一行
MANPATH /usr/local/sarg
```

相信吗？这样就安装完毕了！很快速吧！接下来你要做的就是设定啰！设定之前请先注意您的相关数据，以我为例，我的相关资料为：

- 登录档使用 /usr/local/squid/var/logs/access.log.0 这一个经过 rotation 的档案；
- 计算的暂存档放置在 /tmp 底下；
- 我的 SARG 输出的首页放置在 /var/www/html/sarg 这个『目录』之中；
- 我想要让我的网页以 big5 的字符编码格式输出！

这个时候我只要修改一个档案，也就是 /usr/local/sarg/sarg.conf 即可，内容有点像这样：

```
[root@test root]# mkdir /var/www/html/sarg
[root@test root]# vi /usr/local/sarg/sarg.conf
# 这个档案是 sarg 的设定档，里面已经将整个设定参数讲的很清楚了，
# 你可以依照你的情况来调整这个档案的参数，无论如何，这个档案里面
# 应该至少要有底下这几个数据，其它的请自行使用喔！
language Taiwan_big5
access_log /usr/local/squid/var/logs/access.log.0
title "Squid 使用状态报告"
temporary_dir /tmp
output_dir /var/www/html/sarg
overwrite_report no
mail_utility /bin/mail
topsites_num 100
exclude_codes /usr/local/sarg/exclude_codes
max_elapsed 28800000
charset big5

[root@test root]# /usr/local/sarg/bin/sarg
SARG: 制作报告完成于 /var/www/html/sarg/2003Apr10-2003Apr11
```

试跑一下，嘿嘿！已经成功的输出数据啰！这个时候，请在您的浏览器上面输入 http://your.domain.or.IP/sarg 即可看到刚刚跑出来的资料了！很棒吧！好了，我总不能常常这样手动的跑分析数据吧？OK！那么怎么将这个动作放到 crontab 当中呢？！呵呵！我们底下写了一支 script 来同时跑 pwebstats 及 SARG 这两个玩意儿～这支 script 是长这样的：

```
1. 建立这支程序：我将他取名为 /usr/local/squid/etc/squid.logrotate
[root@test root]# vi /usr/local/squid/etc/squid.logrotate
#!/bin/bash
# 这支程序是要写来做为 squid 的 log files analysis 之用的！
# 执行的方法为 crontab 啰！
# vi /etc/crontab 加入底下这一行：
# 59 23 * * * root /usr/local/squid/etc/squid.logrotate

# 1. parameters settings
```

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin

# 2. stoping and rotating squid
sleep 50s
/usr/local/squid/sbin/squid -k rotate
/usr/local/squid/sbin/squid -k shutdown

# 3. pwebstats processing
/var/www/html/pwebstats/pwebstats -c  \
        /var/www/html/pwebstats/conf/squid-proxy.conf \
        > /dev/null 2>&1

# 4. sarg processing
/usr/local/sarg/bin/sarg > /dev/null 2>&1

# 5. starting squid
sleep 11s
su nobody -c "/usr/local/squid/bin/RunCache &" > /dev/null 2>&1

# 2. 改变档案权限与加入 crontab 排程当中!
[root@test root]# chmod 744 /usr/local/squid/etc/squid.logrotate
[root@test root]# vi /etc/crontab
加入这一行:
59 23 * * * root /usr/local/squid/etc/squid.logrotate
```

这一支程序有什么特点呢？由于我是在 23:59 开始执行，而且执行时期先等待 (sleep) 50 秒钟，且工作完毕后再等 10 秒钟，也就是说，我的 squid 的 log 档案一定会是以天为单位来分隔，所以我们所看到的数据就会『一」天一天』的显示，而不会跨了不同的两天了！我是比较喜欢这样啦！但是每个人的观点不同，您可以自由调配喔！

防火墙的规划

事实上， Proxy 本身就已经可以做为一个防火墙啦！为何还需要针对 Proxy 来进行防火墙的规划？话是这样没错啦！但是我们的 Proxy 是开放 3128 与 3130 这两个 port 啊！您总不能这两个 port 没有开放吧！所以啰，要让您的 Proxy 可以对外面开始服务，就需要启用 3128 这个标准的 Proxy port 啰！所以需要在您的防火墙 scripts 当中加入这一行：

```
/sbin/iptables -A INPUT -p TCP -i eth0  --dport 3128:3130 -j ACCEPT
```

一般来说，这样就可以正常的启动 Proxy 的服务了！那么至于 Proxy 本身提供的防火墙内容呢？呵呵！就利用 acl 配合 http_access 就能够管制使用者的使用空间了！真是相当的具有弹性啊！^_^

NAT 与 Proxy 透过 transparent proxy 设定加快网络传输

让我们现在来想象一个联机状态，就是你有一整组内部网络，而这个内部网络都是透过 NAT 主机联机出去的。那么我们谈过，就是在一个内部网域很大的情况下，使用 Proxy 是一个很不错的选择，因为至少他可以减轻频宽的负荷啊！OK！那么我们就架设 Proxy 好了，不过，遗憾的是，架设 Proxy 的时候，也要使用者在浏览器上面设定了你架设好的浏览器才有用啊！否则那部 Proxy 没有人使用的话，架了也是白搭！好了，那么有没有办法在『使用者不需要在浏览器上面进行任何设定，就可以实现以 Proxy 帮助使用者捉取 WWW 数据 』呢？当然有啦！那就是 Transparent Proxy 啦！也有人翻译成『通透式代理服务器』，为什么这么翻译我也不晓得？？不过，他的原理是这样的：

- 当使用者经过 NAT 服务器来联机进入 Internet 时，假如使用的 Internet 协议为 80 (也就是 WWW) ，那么就将这个要求交给 Proxy 来工作，以达到代理服务器的功能。

呵呵！也就是说，当使用者是经过 NAT 主机联机出去时，只要让 NAT 主机发现『咦！你是要去捉 WWW 的资料对吧！好！那么这个动作由 Proxy 主机帮你搞定！』如此一来，使用者根本就不需要在浏览器上面设定 Proxy 的相关数据，因为这个动作是『由 NAT 主机自己决定的』，所以只要在 NAT 主机上面设定妥当即可，使用者不必设定任何数据呢！呵呵！真是不错！那么要怎么进行呢？只要两个步骤即可：

7. 设定 Proxy 主机：
   设定 proxy 主机当然就是要又修改 squid.conf 啰！而这个设定相当的简单喔！只要几行就可以搞定：

```
[root@test root]# vi /usr/local/squid/etc/squid.conf
# 这里请填入你的 Proxy 主机名称 与 port ！
httpd_accel_host vbird.adsldns.org
# 因为我们是要进行 WWW 的数据快取，所以 port 当然就是 80 啰！
httpd_accel_port 80
# 这个很重要！因为设定 httpd_accel_host 之后， cache 的设定会自动被终止，
# 必须要加上这个设定为 on 之后，才能提供 cache 的功能！
httpd_accel_with_proxy on
httpd_accel_uses_host_header on

[root@test root]# /usr/local/squid/sbin/squid -k reconfigure
```

8.
   总共就是这四行啦！这样就设定好一个 squid 的 transparent proxy 的功能啰！

9. 设定 NAT 主机的 port map ：
   再来让我们到 NAT 主机上面看看先，因为需要将 80 这个 port 交给 Proxy 的 3128 来帮忙协助，所以你的防火墙 script 必须要加入这一段才行：

```
iptables -t nat -A PREROUTING -i eth0 -p tcp -s 192.168.0.0/24 \
--dport 80 -j REDIRECT --to-ports 3128
```

10.
    注意一下，那个 eth0 是『你的 NAT 对内的网络卡装置代号』，至于 192.168.0.0/24 则是你的内部网域，请依照你的主机实际状态来设定喔！

完全不需要怀疑！这样一来，您的 client 端完全不需要进行任何的设定，立刻就可以使用 Proxy 的好处啰！很不错吧！呵呵！

---

squid 的注意事项

使用代理服务器后，浏览国外的网页应该是可以变快的！但是，你要小心几件事：

11. 若 squid 内设定的使用空间满了，则 squid 将不会运作！
12. 若 squid 的纪录文件太大了，则工作效率会变慢！

由于我们已经安装的 pwebstats 了，并且已经设定好了 squid -k rotate 的工作，所以第二个问题并不严重，严重的可能会是第一个问题！所以有可能我们会自行手动的删除 Proxy 的快取目录，如何删除呢？虽然上面已经提过了一些注意事项，这里我们再次的说明吧！

```
1. 停止 squid
[root@test root]# kill -9 `cat /usr/local/squid/var/logs/squid.pid`
(可能会重复做 5 次左右才会完全砍掉！)

2. 删除暂存目录 (这个目录请依您的系统而定！)
[root@test root]# rm -rf /usr/local/squid/var/cache1
[root@test root]# mkdir /usr/local/squid/var/cache1

3. 重建快取目录并重新启动
[root@test root]# /usr/local/squid/sbin/squid -z
[root@test root]# su nobody -c "/usr/local/squid/bin/RunCache &"
```

其它的建议：

13. 关于上层代理服务器：用 cache_peer 设定上层代理服务器的数目不要太多，只要 2-5 个之间就好了，而且上层代理服务器一定要找距离你最近，并且具有较大频宽的主机，如果是在台南，那 proxy.ncku.edu.tw 就是不错的主机，或者向您的 ISP 询问喔；

14. 关于暂存目录的设定：以 cache_dir ufs 设定的目录，最好是单独割出来的约 1-2 GB 的硬盘槽，以我为例，我将另外一台主机的 30GB 的硬盘割两槽给 proxy 用，而每一槽只有 2GB ，分别命名为 proxy1 与 proxy2 ，则可以写成

    ```
    cache_dir ufs /proxy1 2000 16 256
    cache_dir ufs /proxy2 2000 16 256
    ```

    由于分成两槽来存取，所以整体效率上会比较好，但这是针对一般比较大型的代理服务器的设定了，我们这个小主机就不用如此设定(但是效率真的有差哩！)。

15. 善用 acl, always_direct, never_direct：就如同上面提到的，因为你的目的不同，所以会使用到不同的 proxy 作为你的上层代理服务器，如果你发现你的上层代理服务器无法针对你常上的网站来求取资料时，就将那个网站加入你的 always_direct 吧！另外，也可以使用 cache_peer_access 来处理喔！

16. 在 ./configure 的时候增加 --enable-async-io=80 这一个指令：基本上，增加这个指令之后，将可以使您的磁盘多一个 type ，亦即是 aufs ，这个 type 的速度较快！

重点回顾

- 代理服务器 (Proxy) 最大的功能是在代理使用者向 Internet 要求 Web page 的数据，同时达成 Web pages 的快取记录 (Cache) ，以达到假性的频宽节省目的；此外，还可以额外的达成防火墙的功能；
- 目前 Unix Like 的机器中，做为 proxy 功能的服务器软件几乎都是使用 squid ，而 squid 仅需要设定 squid.conf 这个设定档即可使用；
- 设定 Proxy 时，如果能以频宽更大的上层 Proxy 来帮助，将有助于 Client 端浏览速度的提升；
- 以防火墙的功能来说， Proxy 使用应用层的方式来达成防火墙功能，至于 iptables 则是更为底层的 TCP/IP 分析的方式；
- Proxy 对于硬件的要求较高，尤其是硬盘的 partition 与内存，一般来说，一个 cache 目录最好就是一个 partition ，而一个 cache partition 最好容量在 2-4 GB 之间即可；
- transparent proxy 的功能就是可以让 client 端不需要设定浏览器的 proxy 功能，即可进行 proxy 的工作；

参考资源

- 优客笔记本：http://turtle.ee.ncku.edu.tw/~tung/proxy/

- 台北市教育网络中心：http://www.tp.edu.tw/document/squid/index.files/frame.htm
- squid 官方网站：http://www.squid-cache.org/
- squid 说明文件计划：http://squid-docs.sourceforge.net/
- 中山大学资讯工程所：http://www.cc.nsysu.edu.tw/~lmj/Squid.files/frame.htm
- 鹭江国小：http://proxy.lcps.tpc.edu.tw/
- fly 官方网站：http://martin.gleeson.com/fly/index.html
- pwebstats 官方网站：http://martin.gleeson.com/pwebstats/index.html

---

本章习题练习 （ 要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看 ）

- 请说明为何 Proxy 可以提升网络的 WWW 浏览速度？
- 万一 squid 发生了问题，请问我该如何找出问题点？
- 请说明 Proxy 服务器的功能为何？
- 试说明为何 Proxy 服务器可以提升网域之内的网络安全性？

简易 NIS Server 架设

最近更新日期：2003/09/16

有没有想过，如果我有十部 Linux 主机，这十部主机仅负责不同的功能，事实上，所有的主机账号与对应的密码都相同！那么我是将账号与密码分别设定置在十部计算机上面，还是可以透过一部主机做为账号管理的功能，然后其它的主机只要当用户用登入时，就必须要到管理账号的主机上面确认其账号与密码呢？哪一个比较方便而且灵活？当然是找一个账号管理的主机比较方便的多啦！如果有使用者要修改密码，不必要去到十部主机修改密码啦！只要到主要管理主机去修改，其它的主机根本就不需要更动！哈哈！轻松又愉快呢！这个功能的达成有很多的方式，在这里，我们介绍一个很简单的方式，那就是 Network Information Service 这个 NIS 服务器的架设啦！

原理：

在一个大型的网域当中，如果有多部 Linux 主机时，万一要每部主机都设定相同的账号与密码的设定，还真是啰唆。所以，适时的使用一部主要主机（ master server ）管理网域中的所有账号，其它的主机则使用这部主要主机提供的账号与密码来达成让使用者『登入』的作用即可！这样的功能有很多的服务器软件可以达成，这里我们要介绍的则是 Network Information Services, NIS server 这个服务器软件喔！

什么是 NIS 与 NIS 的主要功能：

通常我们都会建议，一部 Linux 主机的功能越简单越好，也就是说，一部 Linux 就专门进行一项服务，这样有许多的好处，这包含功能简单所以系统资源得以完整运用，并且在发生入侵或者是系统产生状况的时候，也比较容易追查问题所在。因此，一个公司内部常常会有好几部 Linux 主

机，有的专门负责 WWW 、有的专门负责 Mail 、有的专门负责 SAMBA 等等的服务。不过，这样虽然有分散风险、容易追踪问题的好处，不过，由于主机数量多了，然而因为是同一个公司里面，所以，事实上所有的 Linux 主机的账号与密码都是一样的！哇！如果公司里面有 100 的人的话，那么我们就需要针对这么多部的主机去设定账号密码了！而且，如果未来还有新进员工的话，呵呵！那么光是设定密码就会使系统管理员抓狂了！

这个时候，如果我们换一个角度来思考：如果我设计了一部专门管理账号与密码的主机，而其它的 Linux 主机当有客户端要登入的时候，就必须要到这部管理密码的主机来查寻使用者的账号与密码，如此一来，哈哈！我要管理所有的 Linux 主机的账号与密码，只要到那部主要主机上面去进行设定即可！包括新进人员的设定，反正其它的 Linux 主机都是向他查寻的嘛！没错！真是好～这个就是 Network Information Service, NIS 主机的主要功能啦！

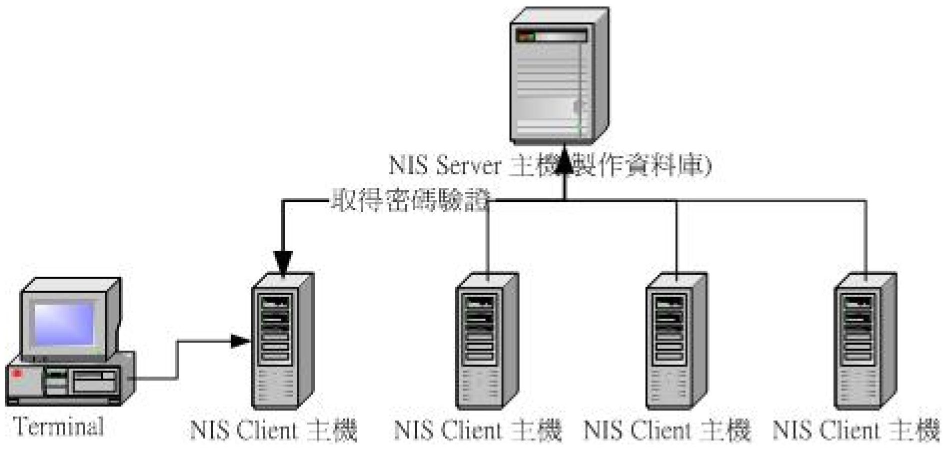
事实上， Network Information Service 最早应该是称为 Sun Yellow Pages (简称 yp)，也就是 Sun 这家公司出的一个名为 Yellow Pages 的服务器软件，请注意， NIS 与 YP 是一模一样的咚咚喔！这个 Yellow Pages 名字取的真是好！怎么说呢？知道黄页(Yellow Pages)是什么吗？没错！就是我们家里的电话簿啦！今天如果你要查寻一家厂商的电话号码，通常就是直接去查黄页上面的纪录来取得电话号码啊！而这个 NIS 也一样，当使用者要登入时， Linux 系统就会到 NIS 主机上面去找寻这个使用的账号与密码信息来加以比对，以提供使用者登入之用的检验啊！很棒吧！ ^_^

那么 NIS 主机提供了哪些信息呢？还记得账号与密码放置在哪里吧？！那么 NIS 就是提供那些数据啦！有底下这些基本的数据提供给 Client 端喔：

- 登入账号/密码/家目录：就是 /etc/passwd 这个档案
- 群组信息：就是 /etc/group 这个档案
- 相关主机名称与 IP：就是 /etc/hosts 这个档案。

NIS 的运作流程

事实上，NIS 的运作流程一点也不困难。如果在一个不是很大的网域当中，那么大约会有一部 NIS Server ，并且同时有很多部的 NIS Client 才对！这里我们不谈 NIS Server 的 Primary 与 Slave 架构，仅谈 NIS Server 与 NIS Client 的架构，整体架构有点像底下的图示：



就如同上面图示的模样。我们已经晓得 NIS 提供的其实就是 /etc/passwd, /etc/group 以及 /etc/hosts 等 ASCII 格式档案的信息，而 NIS Server 会将前述几个 ASCII 档案内容复制成为 DBM 数据库格式的档案，当用户藉由个人计算机联机到 NIS Client 主机尝试登入时，NIS Client 将会到 NIS Server 去查寻该用户的账号与密码，以做为用户登入验证的依据。

- NIS Server 将自己系统内的 /etc/passwd, /etc/group, /etc/hosts 等制作成为 DBM 的数据库格式档案；
- NIS Client 若有用户登入的要求时，会前往 NIS Server 搜寻数据库里面的数据做为验证之用。
- 每次更动 NIS Server 上面的用户数据时，则 NIS Server 需要重新制作 DBM 数据库档案才行！

这样可以了解吗？！很简单的啦！不过，需要特别留意的是，我们需要设定的有：

- NIS Server 端；
- NIS Client 端。

设定方面需要两者的设定喔！

NIS 与 RPC 的关系

还记得另一个 Sun 公司开发的服务器软件 NFS 吗？他是由 RPC (Remote Procedure Call) 所统一管理的。呵呵！我们这个 NIS 也是使用 RPC 来管理的喔！所以啰！您最好回到前面 NFS 的章节去瞧一瞧 RPC 的一些相关说明比较好喔！那个 RPC 就是我们常常见到的 Portmapper 啦！也就是 sunrpc (port 111) 啰！在 NIS 里面，我们不但需要启动 portmap ，还需要启动另一个玩意儿，那就是由 super daemon 管理的 time 与 time-udp 这两个宝贝蛋了！嗯！底下我们将会来探讨一下各个套件啰！

NIS Server 的 master 与 slave 架构

刚刚我们仅提到只有一部 NIS Server 在整个网域之中，这是一般比较小型的网域常见的方法。万一，如果我们的网域里面有几乎 100 部以上的主机呢？而且每一部的流量还真的很大的时候，这个时候，只有一部 NIS Server 可能无法提供快速的数据查寻与响应的状态！这个时候就需要 NIS Sever 的 master 与 slave 的架构了。

还记得在 DNS 主机架设当中，我们曾经提过关于 master 与 slave 的关系吧？！就是 slave 主要是藉由将来自 master 主机的数据加以更新到自己的数据库当中，并且提供与 master 相同的查寻功能！这个 NIS 的 master 与 slave 架构则完全相同！

- NIS Server 的 master 先将自己的账号、密码相关档案制作成为数据库档案(database file)；
- NIS Server 的 master 将自己的数据库档案传送到 slave 上面；
- NIS Server 的 slave 接收来自『信任的 NIS Server master 主机』的数据后，更新自己的数据库，使自己的数据库与 master 主机的数据同步；
- 网域当中的所有 NIS Client 查寻 NIS Server 时，会找寻『最先响应的那一部 NIS 主机的数据库内容』。

所以，我们可以知道的是，NIS 的 master 与 slave 架构主要在分散查寻 NIS 时候的主机负荷，因此，除非您的网域真的很大，否则是没有必要架设 NIS Slave 与 master 的架构的啦！底下我们没有架设 master 与 slave 喔！只有一部主要的 master 而已啦！

套件安装

基本上，NIS 建议直接使用原版光盘上面给我们的 RPM 来安装即可!但是需要安装哪些套件呢？您至少需要底下几个套件才行：

- yp-tools ：提供 NIS 相关的查寻指令功能
- ypbind ：提供 NIS Client 端的设定套件
- ypserv ：提供 NIS Server 端的设定套件
- portmap ：就是 RPC 一定需要的数据啊！

我是在 Red Hat 系统上面使用的设定，所以档名是这样的一个模样，你可以使用『 rpm -qa | grep yp 』来检查一下是否真的有安装这些个套件才行！不过，为什么 NIS Server 的套件名称会是 yp 呢？还记得我们在上面提到的信息吗？ NIS 最早的名称是 Sun Yellow Pages ，所以啰，套件名称才会是 yp 啊！ ^_^！这样好记多了吧！闲话不多说，马上来进行设定吧！

Server 端设定：

终于来到了设定的地方了， NIS 的设定与 NFS 的设定有点小小的相同之处，就是他的设定『粉简单！』的啦！架设他吧！

NIS Server 的结构

NIS Server 主要以 ypserv 这个套件提供的数据来进行设定的，他主要的内容有：

- /etc/ypserv.conf ：就是主要的设定档了
- /usr/sbin/ypserv ：主要的服务(daemon)执行档
- /usr/sbin/rpc.yppasswdd：RPC 的服务啰！
- /usr/sbin/rpc.ypxfrd ：同样的， RPC 的服务啰！
- /usr/lib/yp/ypinit ：建立 NIS 数据库的执行程序

所以，事实上我们最重要的就是设定 ypserv.conf 这个档案而已啦！至于 RPC 的设定，就直接启动他即可！另外，还有 yp-tools 会提供的相关数据喔：

- /usr/bin/yppasswd ：更改你在 NIS database (NIS Server 所制作的数据库) 的密码
- /usr/bin/ypchsh ：同上，但是是更改 shell
- /usr/bin/ypchfn ：同上，但是是更改一些使用者的讯息！

NIS Server 设定流程

开始设定吧！在我的系统当中，假定我的网络状况如下：

- 网域为 192.168.10.0/24
- NIS Server 的 IP 为 192.168.10.30，对应的主机名称为 server.cluster
- NIS 的领域名称设定为 cluster

在 NIS Server 端以 root 身份登入后，进行下面的工作：

12. 启动 portmap 并设定开机时启动：
    这个应该不难吧！使用：

```
[root@test root]# /etc/rc.d/init.d/portmap start
[root@test root]# netstat -tl
Active Internet connections (only servers)
```

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:sunrpc                *:*                     LISTEN
# 如果看到 sunrpc 的话，就表示启动成功了！
[root@test root]# chkconfig --level 35 portmap on
# 上面这一行在设定 portmap 在 run-level 为 3, 5 的时候就开机时启动！
```

13. 很简单吧！这样 portmap 就启动了！

14. 启动 time 与 time-udp ：
    由于 time 与 time-upd 是在 NIS 运作时所需要的 daemon ，所以也必须要启动他啦！启动的方式也很简单，就是利用 xinet 这个 super daemon 来进行即可！

```
[root@test root]# vi /etc/xinetd.d/time
# 找到底下这一行：
disable = yes
# 将他改成
disable = no
# 储存后离开
[root@test root]# vi /etc/xinetd.d/time-udp
# 同样的将 disable = yes 改成 disable = no 即可！

[root@test root]# /etc/rc.d/init.d/xinetd restart
[root@test root]# netstat -utl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:time                  *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
udp        0      0 *:time                  *:*
udp        0      0 *:sunrpc                *:*
```

15. 注意喔！目前至少也要有上面四个 LISTEN 的 port 才行喔！分别是 UDP 与 TCP 封包的啦！

16. 建立 NIS 的领域名称 (nisdomainname)
    在 NIS 的系统当中，他的领域名称 (domain name) 是与 DNS 没有绝对关系的，由于这个领域名称会常被使用到，因此我们需要将他建立起来！建立起来的方法很简单，直接执行一个指令，并修改一个档案即可！

```
1. 建立 NIS 的领域名称 (我这里是设定为 cluster )：
[root@test root]# nisdomainname cluster
[root@test root]# vi /etc/rc.d/rc.local
# 将底下这一行加入这个档案的最后一行内：
 /bin/nisdomainname cluster
```

```
2. 设定好网络参数：
[root@test root]# vi /etc/sysconfig/network
# 加入底下这一行：
NISDOMAIN=cluster
```

17.
18. 设定 ypserv 的设定档： ypserv.conf
这个设定档的内容其实也是很简单，大概只有几行而已，设定的主要语法为：

```
<设定项目>:<设定项目的值>
```

19. 我们先谈一谈他里面的几个设定细项：

```
[root@test root]# vi /etc/ypserv.conf
files: 30
# 这说的是『有多少数据库档案(database file)会被先读进高速缓存当中』
# 的意思，一般来说， 30 是已经很足够的数值了，不需要更动他；

trusted_master: your.master.servers.name
xfr_check_port: yes
# 上面这两个都仅与 Master + Slave 架构有关的设定值，一般来说，
# 只有一部主要 NIS Server 的系统中是用不到这两个设定值的！
# 如果你的 NIS 是 slave 的架构，那么需要指定一部 master 做为数据库内容的
# 同步时候的主机，那就是 trusted_master 的设定内容啰！
# 如果没有 master/slave 架构时，那就不需要 trusted_master 这个设定了！
# 至于 xfr_check_port 则是指定 master 与 slave 是否都要以 < 1024
# 以下的 port 来进行沟通的讯息！通常预设就是 yes ，不需要更动他！

# <主机名称/IP>:<网域名称>:<数据库类别>:<安全性>
# 这个是这个档案里面最重要的部分了！主要在设定安全性的方面，
# 可以设定多行，而是否能够通过的规则是『一行一行检查』的方式！
# 所以这里的设定应该是：先开放要开放的网域，然后全部都关闭！
# 先谈一谈各个相关的项目：
# 1. 主机名称/IP：这里可以这样设定：192.168.1.0/255.255.255.0
# 2. 网域名称：通常都设定成为 * 即可！
# 3. 数据库类别：可以使用 * 来表示所有的数据库！
# 4. 安全性：主要有三种参数：
#            none ：无论如何就是可以无条件进入本机；
#            port ：仅允许 < 1024 以下的 port 进入；
#            deny ：无论如何就是关闭不让人家登入主机！
# 由于我是允许 127.0.0.0/255.0.0.0 以及 192.168.10.0/255.255.255.0 进入，
# 其它的都关闭！所以我可以这样设定：
```

```
127.0.0.0/255.255.255.0    :  *  :  *  :  none
192.168.10.0/255.255.255.0:  *  :  *  :  none
*                          :  *  :  *  :  deny
# 但是因为 /etc/shadow 里面的档案总不好让人看到吧！而又由于 Linux
# 系统当中，只有 root 可以启用 < 1024 以下的 port ，因此，更安全的设定，
可以这样做：
127.0.0.0/255.255.255.0    :  *  :  *  :  port
192.168.10.0/255.255.255.0:  *  :  *  :  port
*                          :  *  :  *  :  deny
# 三行也就够了！
# 无论如何，如果您想要让您的 NIS Server 运作的较为快速，并且安全性上面
# 没有太多的考虑(内部网域时！)，那么使用 none 是一个不错的主意！
```

20.
21. 建立网络信任群组：
这个 /etc/netgroup 档案可以记录在我们网域里面被信任的群，这个档案的内容当中，每一行都有三个字段，分别以逗号『,』隔开，意义为：

```
<host>,<user>,<domain>
主机,使用者账号,领域名称
```

22. 事实上，如果这个档案是『空的』的话，那么代表着『全部的主机、账号与领域名称都接受』的意思，因为我们已经在 /etc/ypserv.conf 里头设定好了关于安全的项目了，所以这个档案只要建立即可(本来是不存在的！)：

```
[root@test root]# touch /etc/netgroup
```

23.
24. 启动 ypserv 这个 daemon ，并且设定开机时启动：
好了！都设定完成之后，在接下来自然就是要启动了！启动有两个 daemons ，启动的方式为：

```
1. 启动啰！
[root@test root]# /etc/rc.d/init.d/ypserv    start
[root@test root]# /etc/rc.d/init.d/yppasswdd start

2. 观察一下是否真的有动作？
[root@test root]# rpcinfo -u localhost ypserv
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
[root @test root]# rpcinfo -u localhost yppasswdd
program 100009 version 1 ready and waiting
# 这个 rpcinfo 就是在观察与 RPC Server 有关的 program 目前的状况！
```

```
# 因此可以用他来观察你的 ypserv 喔!

3. 设定开机时启动:
# 你可以使用 ntsysv ，这里我们使用 chkconfig 啰!
[root@test root]# chkconfig --level 35 ypserv on
[root@test root]# chkconfig --level 35 yppasswdd on
```

25.

26. 制作数据库、并重新启动 ypserv 与 yppasswd ：
好了，既然 NIS Server 主要是要提供数据库给大家参考用的，所以当然要制作数据库啰！然后，我们又将这些数据库读入快取当中，所以数据库制作完毕之后，一定要重新启动 ypserv 与 yppasswdd 才行！

```
1. 制作数据库:
[root@test root]# /usr/lib/yp/ypinit -m
At this point, we have to construct a list of the hosts which will run NIS
servers.  server.cluster is in the list of NIS server hosts.  Please continue to add
the names for the other hosts, one per line.  When you are done with the
list, type a <control D>.
        next host to add:  server.cluster
        next host to add: <==在这里按下[ctrl + d]跳出
The current list of NIS servers looks like this:

server.cluster

Is this correct?  [y/n: y]  y
We need a few minutes to build the databases...
Building /var/yp/cluster/ypservers...
Running /var/yp/Makefile...
gmake[1]: Entering directory `/var/yp/cluster'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating hosts.byname...
Updating hosts.byaddr...
Updating rpc.byname...
Updating rpc.bynumber...
Updating services.byname...
Updating services.byservicename...
Updating netid.byname...
Updating protocols.bynumber...
Updating protocols.byname...
Updating mail.aliases...
```

```
gmake[1]: Leaving directory `/var/yp/cluster'
server.cluster has been set up as a NIS master server.
Now you can run ypinit -s server.cluster on all slave server.

# 这个动作是每次修改使用者数据后一定要做的动作！，就是重新制作数据库，
# 然后并且需要重新启动 ypserv 与 yppasswdd 喔！

2. 重新启动服务：
[root@test root]# /etc/rc.d/init.d/ypserv    restart
[root@test root]# /etc/rc.d/init.d/yppasswdd restart
```

27. 这个动作的重点是在 /var/yp 这个目录当中，制作了多个的等待 NIS Clients 查寻的数据库！请注意的是，每次在 NIS server 上面更动使用者的数据时，一定需要重新做这个步骤喔！

这样 Server 的部分就设定妥当了！如果您还想要玩一玩 master 与 slave 的架构的话，那就请参考：
NIS HOW-TO：http://www.linux-nis.org/nis-howto/HOWTO/index.html

---

Client 端设定

设定完了 Sever 之后， NIS Client 也需要设定喔！(注：在 NIS clients 主机记录的登入者的信息中，仅记录 UID 大于 500 以上的使用者喔！因为小于 500 以下的 UID 都是预设给系统使用的，因此是预设不开放给 NIS 来查寻，自然也就不会被写入 NIS 数据库档案当中了！)

NIS Client 的结构

还记得上面提过的，NIS Client 需要的套件是：

- ypbind
- yp-tools

至少也要这两个套件才可以喔！至于相关的设定档为：

- /etc/yp.conf ：设定 NIS Server 的主机名称与领域名称
- /etc/hosts ：至少需要设定 NIS server 主机 IP 对应的主机名称喔！
- /etc/passwd ：指定需要查寻的是什么；
- /etc/nsswitch.conf ：指定要使用什么 daemon 查寻账号与密码。

大致上就是如此啦！我们要设定的信息也就是如同上面的档案啰！好！设定吧！

NIS Client 的设定流程

请留意的是，底下的设定都是在 Client 端喔！不要在主机端作这些设定了！^_^

7. 启动 portmap 并设定开机时启动：
   不论是 RPC Server 还是 RPC Client ，反正只要是 RPC 的相关服务要应用，就一定要有 portmap 的辅助才行！所以，启动并设定开机时启动吧！

```
[root@client root]# /etc/rc.d/init.d/portmap start
[root@client root]# netstat -tl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:sunrpc                *:*                     LISTEN
如果看到 sunrpc 的话，就表示启动成功了！
[root@test root]# chkconfig --level 35 portmap on
上面这一行在设定 portmap 在 run-level 为 3, 5 的时候就开机时启动！
```

8. 很简单吧！这样 portmap 就启动了！

9. 设定 NIS Server 的 IP 对应主机名称：
   请看上面 Server 设定之前的假设，我的 NIS Server 的 IP 与主机名称记录在 Client 端的 /etc/hosts 上面：

```
[root@client root]# vi /etc/hosts
# 要有底下这一行喔！
192.168.10.30 server.cluster
```

10.
11. 设定 NIS 的 domain 与 NIS 的主机：
   NIS Server 与 Client 的 NIS domain 一定要相同，所以我们必须要花一点时间来将这个咚咚搞定：

```
1. 建立 NIS domain name：
[root@client root]# nisdomainname cluster
[root@client root]# vi /etc/rc.d/rc.local
# 加入底下这一行：
/bin/nisdomainname cluster
[root@client root]# vi /etc/sysconfig/network
```

```
# 加入底下这一行：
NISDOMAIN=cluster

2. 建立 NIS 查寻的主机名称
[root@client root]# vi /etc/yp.conf
# 加入这两行：
domain cluster
ypserver server.cluster
# 还是要记得，那个 cluster 是你的 NIS 的 domain ，至于 server.cluster
# 则是 NIS Server 的主机名称，我这里是使用内部私有 IP ，所以名称可以随便
# 我喜欢来选择的喔！
```

12.

13. 修改密码验证的方式：

    密码验证的方式是一定要修改的，不然你的系统怎么知道要去哪里使用什么方式查寻账号、密码数据呢？！所以您至少需要更改两个档案喔！

```
1. 密码文件的修改：
[root@client root]# vi /etc/passwd
# 还记得这个档案吗？这个档案总共有七个字段，而每个字段都以分号『:』隔开，
# 相关的信息请参考基础学习篇里面的账号管理章节。由于我们要将数据
# 设定以 NIS Server 的数据库来验证，因此，
# 在这个档案的最后面加入这一行：
+::::::
# 注意喔！在 + 之后连续加六个『 : 』，并且中间没有空格符！

2. 查寻密码的程序：
# 因为我们有很多方式来查寻密码，需要修改 /etc/nsswitch.conf 这个档案才行
[root@client root]# vi /etc/nsswitch.conf
# 找到相关的参数，并改成底下这样：
passwd:     files nis nisplus
shadow:     files nis nisplus
group:      files nis nisplus
hosts:      files nis dns
# 这个档案在设定一些信息的查寻程序！那个 files 指的是本机的相关档案，
# 至于 nis 则是透过 NIS 来进行查寻，至于 nisplus 则是 NIS+ 这是比较新
# 版的 NIS 啦！不过，官方网站上面说，目前这个发展的计划已经暂停了！
# 1. passwd: 就是使用者相关信息查寻，分别为 /etc/passwd, nis 与 nisplus
# 2. shadow: 就是使用者密码的查寻， /etc/shadow, nis 及 nisplus
# 3. group:  就是使用者的群组信息查寻， /etc/group, nis 及 nisplus
# 4. hosts:  就是主机名称与 IP 对应的查寻，/etc/hosts, nis 及 /etc/resolv.conf
```

14.

15. 启动 ypbind 与设定开机启动：
    这样几乎就设定完成了！而我们前面说过， NIS Client 也需要启动 ypbind 这个 daemon 的，所以就给他启动吧！

```
1. 直接手动启动 ypbind 吧：
[root@client root]# /etc/rc.d/init.d/ypbind start
[root@client root]# rpcinfo -p localhost
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100007    2   udp    735  ypbind
    100007    1   udp    735  ypbind
    100007    2   tcp    738  ypbind
    100007    1   tcp    738  ypbind
# 至少要有上面几个信息才是对的喔！不过，需要记得的是，那个 port number
# 是系统随机启动的，所以 port number 每次都会不太一样啊！
# 记得要经常用 rpcinfo 去检查一下 RPC Server 相关的服务才行！

2. 设定开机时启动：
[root@client root]# chkconfig --level 35 ypbind on
```

16.

基本上，上面的动作就已经设定好了一部 NIS Client 主机了！而且已经可以跑啰！不过，我们毕竟不知道到底目前我们的 NIS Client 主机使用的 NIS Server 主机里面的哪些数据，并且如何去确认我们的数据库与设定值都没有问题呢？！呵呵！所以底下我们就来谈一谈那个 yp-tools 提供的相关好用的工具程序来检验啰！

NIC Client 端检验 NIS 设定：yptest, ypwhich, ypcat

设定好了 NIS Client 之后，先以 netstat 与 rpcinfo 检验一下是否成功的启动之后，接下来就是要测试到底我们的 NIS Client 与 NIS Server 之间沟通的情况如何了！此时就需要使用到 NIS 提供的 yp-tools 这个套件，里面的几个小小的执行程序来动作了：

- yptest ：主要在测试 yp 的设定内容、数据库内容等等所有 NIS 相关的数据测试；
- ypwhich：主要在测试 NIS Client 与 Server 之间沟通的数据库 (database) 到底是哪几个档案；
- ypcat ：主要在取得 NIS Server 上面的使用者密码信息！

分别谈一谈每个程序的用途与说明吧！

---

yptest

```
[root@client root]# yptest
Test 1: domainname
Configured domainname is "cluster"

Test 2: ypbind
Used NIS server: server.cluster
....
....
Test 8: yp_maplist
rpc.bynumber
rpc.byname
hosts.byaddr
hosts.byname
group.byname
passwd.byname
ypservers
passwd.byuid

Test 9: yp_all
test test:dkoUW2XHV30sEV5gLM4NapyuhBcpVs.:500:500::/home/test:/bin/bash
```

看到了吗？会有很多的资料一项一项的去测试，测试的结果都会显示在屏幕上面，最好都没有问题之后再开始 NIS Client 的服务吧！ ^_^

---

ypwhich

```
[root@client root]# ypwhich
server.cluster
[root@clent root]# ypwhich -x
Use "ethers"    for map "ethers.byname"
Use "aliases"   for map "mail.aliases"
Use "services"  for map "services.byname"
Use "protocols" for map "protocols.bynumber"
Use "hosts"     for map "hosts.byname"
Use "networks"  for map "networks.byaddr"
Use "group"     for map "group.byname"
Use "passwd"    for map "passwd.byname"
```

单纯使用 ypwhich 的时候显示的是『NIS Client 的 domain』名称，而当加入 -x 这个参数时，则是显示『NIS Client 与 Server 之间沟通的数据库有哪些？』由上面我们可以很清楚的就看到相关的档案啦！这些数据库档案则是放置在我的 NIS Server 的 /var/yp/cluster/* 里面啰！

---

ypcat

```
[root@client root]# ypcat -x
Use "ethers"    for map "ethers.byname"
Use "aliases"   for map "mail.aliases"
Use "services"  for map "services.byname"
Use "protocols" for map "protocols.bynumber"
Use "hosts"     for map "hosts.byname"
Use "networks"  for map "networks.byaddr"
Use "group"     for map "group.byname"
Use "passwd"    for map "passwd.byname"
# 主要的功能就是『列出数据库』啰！与 ypwhich -x 相同功能！
# 所以我们有 ethers，aliases......passwd 等数据库名称与文件名！

[root@client root]# ypcat [数据库名称或功能]
# 这个指令可以用来取得 NIS Server 上面各个数据库的内容！
# 举例来说，我们想要知道 passwd ( 密码数据 ) 的所有使用者内容，就需要：
[root@client root]# ypcat passwd  (或 ypcat passwd.byname )
test:dkoUW2XHV30sEV5gLM4NapyuhBcpVs.:500:500::/home/test:/bin/bash
# 如果是想要知道 hosts 的内容 (NIS Server 主机上面 /etc/hosts 的内容)：
[root@client root]# ypcat hosts
127.0.0.1       localhost       localhost.localdomain
192.168.10.30   server.cluster
# 反正就是加上数据库，你就可以取得 NIS server 主机上面的数据库内容啦！
```

这三个指令在进行 NIS Client 端的检验时，是相当有用的喔！不要忽略了他的存在啊！尤其是刚架设好 NIS Client 时，一定要使用 yptest 去检查看看有没有设定错误喔！根据屏幕显示的讯息去一个一个校正错误才行啊！

修改使用者密码 (需要有 root 身份)： yppasswd, ypchfn, ypchsh

好了，既然 NIS Client 已经可以正式的来 run 了，那么还可能有什么大问题呢！？最大的问题在于....我能不能在 NIS Client 端修改各个账号的密码呢？答案是『能！』但是不怎么方便～因为，我们要修改的是 NIS Server 端的数据库喔！也就是说，我们在 NIS Client 端登入之后，要修改自己这个账号的密码，其实改到的是 NIS Server 的数据库密码啊！而要修改数据库密码时，需要使用 root 的身份，所以一定需要 root 的密码～如此一来，实在是不太方便～如果真的要修改的话，那么可以使用底下三个小指令来进行修改，不过，不怎么建议这样做就是了！

- yppasswd ：与 passwd 指令相同功能；
- ypchfn   ：与 chfn 相同功能；
- ypchsh   ：与 chsh 相同功能。

无论如何，我是不太建议大家使用这些指令去修改数据库的内容啦！比较建议这样做：

23. 登入到 NIS Server 主机里面去，进行 useradd 或者是 passwd 修改账号与密码等等的更动；
24. 使用 /usr/lib/yp/ypinit -m 重新制作数据库档案！

这样就 OK 啦！比较简单啦我想～至于上面三个指令，请使用 Linux 的好朋友 man 来查看一下吧！ ^_^

主机进阶设定

- NIS 与 NFS 的结合设定：

  不晓得您有没有发现一件事情啊！那就是：我们的 NIS Server 设定的使用者家目录是在 /home 底下，例如 test 这个人的家目录在 /home/test(这个目录在 server.cluster 这部主机上面才有)，问题是，当我们登入 NIS Client 主机时，那么我们取得的家目录数据还是在 /home/test ，问题是，NIS Client 主机并没有 /home/test 这个目录啊：

  - test 这个 User 是在 server 上面建立的，所以有 /home/test 这个目录；
  - 在 NIS Client 上面没有真正的 test 这个账号，因为他是由 NIS server 上面取得的，所以自然也就没有 /home/test 这个目录在 NIS client 上面。

  这样会造成什么问题呢？呵呵！就是你的 test 这个使用者，登入 NIS client 的时候，『会找不到自己的家目录』！啊！真是糟糕～而且，因为我们的 NIS client 可能有很多部，要是每次登入 NIS Clients 主机的时候，所拥有的家目录都是个别 NIS client 上面的目录，那么就没有达到 NIS 的功能啦！您说是吧！所以，如果你需要『登入的每个 NIS Client 所拥有的家目录都是相同的！』的一个情况，呵呵！就可以使用 NFS 来加以设定啦！详细的 NFS 设定我们之前已经提过了，这里不在赘言，单纯谈一下简单的设定技巧：

  - 在 NIS Server 上面开放 /home 这个目录出来；
  - 在 NIS Client 上面， mount NIS 主机的 /home 到自己的 /home 里面去！
  - 如此一来，不论登入哪一部 NIS Server 或 client ，使用者都是进入到 NIS Server 的 /home 里面的家目录啰！

  设定的方法也不难，我们就简单的谈一谈吧！

```
1. 设定 NIS Server 主机的 NFS 开放目录：
```

```
[root@test root]# vi /etc/exports
/home  192.168.10.0/24(rw,async,no_root_squash)

[root@test root]# exportfs -rv
exporting 192.168.10.0/24:/home

[root@test root]# /etc/rc.d/init.d/nfs start
Starting NFS services:                                    [  OK  ]
Starting NFS quotas:                                      [  OK  ]
Starting NFS daemon:                                      [  OK  ]
Starting NFS mountd:                                      [  OK  ]

[root@test root]# chkconfig --level 35 nfs on

2. 设定 NIS Client 的 mount 数据!
# 先以 root 的身份登入到 NIS Client 主机上面：
[root@client root]# mount -t nfs 192.168.10.30:/home /home
# 如果没有问题了，就将上面这一行加入 /etc/rc.d/rc.local 当中吧！
```

这样一来，您的 NIS Clients 就具有和 NIS Server 主机一模一样的家目录了！现在您可以立刻登入看看喔!

- 不过，很可惜的是，目前安装妥当的系统当中， NIS 并没有办法让 SSH 顺利的登入的!这牵涉到整个 key pair 的问题，比较麻烦，或许可以藉由 NIS+ (nisplus) 来克服这个问题，网络上目前有相当多的讨论在讨论这个情况，由于 NIS plus 与 NIS 功能差不多，实在不想再 NIS 上面又架设一部 NIS+，因此目前倾向于不要管 ssh 这种登入的方法，反正做了 NIS 在内部网络当中，最主要的功能其实是在于 R shell 这个比较危险等级的 shell 说!

防火墙的规划

又来到了防火墙的规划了！要注意的是，我们的 NIS 与 NFS 都是使用 RPC Server 的，所以啰，都可以直接管制 111 这个 port 即可！能够直接以 iptables 管理 111 这个 port ，例如仅允许 192.168.10.0/24 这个网域进来的话，可以在你的防火墙规则上面加上：

```
/sbin/iptables -A -s 192.168.10.0/24 --dport 111 -j ACCEPT
/sbin/iptables -A --dport 111 -j DROP
```

此外，你也可以使用 TCP_Wrappers 来掌管喔：

```
[root @test root]# vi /etc/hosts.allow
portmap: 192.168.10.0/255.255.255.0

[root @test root]# vi /etc/hosts.deny
portmap: ALL
```

至于其它的管理，嘿嘿！就得靠您自己发挥创意啰！ ^_^

重点回顾

- Network Information Service (NIS) 也可以称为 Sun Yellow Pages (yp) 主要是负责在网域当中帮忙 NIS Client 端查寻账号与密码以及其它相关网络参数的服务，只是在 Linux 上面称为 NIS 而在 Sun Unix 上面称为 Yellow Pages 而已～
- 网域当中有很多 linux 主机时，可以让一台 Linux 主机做为 NIS Server 主机，负责整个网域当中账号与密码的数据库档案制作；至于其它的 Linux 主机则设定为 NIS client 端，当有用户要进行登入的行为时， NIS client 会前往 NIS server 搜寻数据库里面记录的内容，以做为用户登入的验证信息。
- 不论是 NIS 或者是 NFS 都是藉由 RPC Server 所启用的，因此，都可以使用 rpcinfo 来查寻 NIS 是否已经启动，以及该 daemon 是否已经向 portmapper (RPC server) 注册了！
- NIS 使用的套件就是 yp 这个套件，主要分为两部份， ypserv 用在 NIS Server，至于 ypbind 与 yp-tools 则用在 NIS Client 上面。
- NIS server 其实就是提供本身的 /etc/passwd, /etc/shadow, /etc/group, /etc/hosts 等账号密码数据，以及相关的网络参数等，以提供网域当中 NIS Client 的搜寻之用；
- 在 NIS Server 的设定当中，最重要的一个步骤就是将账号、密码、网络参数等 ASCII 格式档案转成数据库档案(database file)，以提供 NIS client 的查寻！而启动 ASCII 转成 database 的程序可以使用 /usr/lib/yp/ypinit -m 或者到 /var/yp 底下执行 make 均可。
- NIS client 端的设定当中，最重要的为 /etc/passwd 里面的设定，需要让 NIS Server 的数据加在 /etc/passwd 后面；
- NIS client 端的设定当中，另一个重要的设定档为 /etc/nsswitch.conf ，里头设定了相当多的数据查寻程序。

参考资源

- Study Area 之 NIS 服务器架设：http://www.study-area.org/linux/servers/linux_nfs.htm
- NIS 官方网站：http://www.linux-nis.org/
- NIS HOW-TO：http://www.linux-nis.org/nis-howto/HOWTO/index.html

---

本章习题练习

- 请简单说明 NIS server 的功能与工作流程
- 请简单说明 NIS Server/client 的架构
- NIS 启动之前需要先启动那个服务，否则就无法启动成功 (提示：RPC Server)
- 我的 NIS 网域名称为 bird ，另外，我主机的 IP 与主机名称为 192.168.5.1/bird.nis.org ，请问要这些信息需要设定在 NIS Server 的哪些档案之内？
- /etc/nsswitch.conf 的功能为何？如果我想要让密码查寻先本地的密码文件，再查寻 NIS ，需要如何设定？
- NIS Server 将密码等档案做成数据库以提供 NIS client 来查寻，那么请问使用什么动作后，可以将密码档案转成 NIS 的数据库格式档案？
- 如果我想要增加网域当中一个新的账号：newaccount，并且这个 newaccount 可以让 NIS Client 查寻到他的账号与密码，需要进行哪些步骤？
- 实作范例题：底下是我的网域参数特征：
  network/netmask:192.168.1.0/255.255.255.0
  NIS server : 192.168.1.100 (hostname: server.nis.test)
  NIS cient: 192.168.1.200 (hostname: client1.nis.test)
  NIS domain name: nis.test
  利用上面的参数来设定 NIS 架构，请一步一步的写下你的设定。
- 承上题：如果我的网域太大了，所以有一部 NIS slave 主机，这部主机的 IP 为 192.168.1.50 ，请问这部主机该如何设定？
  前往参考用解答

简易 NTP 服务器设定

最近更新日期：2003/08/21

很多时候由于计算机硬件的问题，所以我们都会调整一下时间，好让计算机系统的时间可以一直保持正确的状态。而既然要调整时间，那么自然就会有一个让我们可以对照着来调整时间的『准确时间』咯！在实际生活中，我们可以透过电视台、广播电台、电话等等来调整我们的手表，那么如果是在网络上呢？该如何让我们的主机随时保持正确的时间信息？！另外，整个地球被切分成为 24 个时区，那么什么是 GMT(格林威治时间)，我们所在的时区又是哪一区呢？！让我们来简单的谈一谈吧！

原理

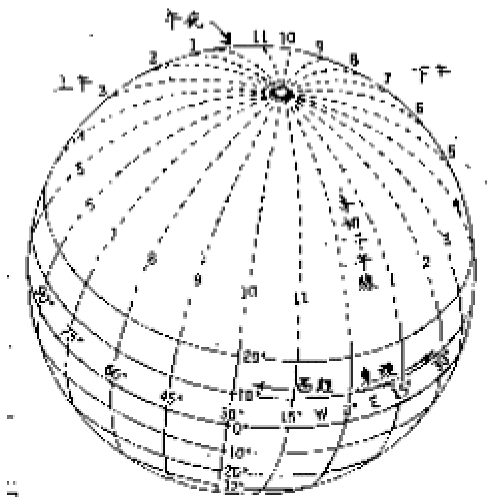
『时间』对于现在人来说，是很重要的！因为时间就是金钱啊！在 Internet 上面，时间同样也是挺重要的！有些计算机需要时间同步才能够正常的运作哩！所以说，时间真的是挺重要的！在开始 NTP 服务器的介绍之前，我们先来简单的谈一谈关于『时区』的概念吧！

什么是时区？全球有多少时区

因为地球是圆的，所以同一个时刻，在地球的一边是白天，一边是黑夜。而因为人类使用一天 24 小时的制度，所以，在地球对角的两边就应该差了 12 的小时才对。由于同一个时间点上面，整个地球的时间应该都不一样，为了解决这个问题，所以可以想见的，地球就被分成 24 个时区了！

那么这 24 个时区是依据什么来划分的呢？由于地球被人类以『经纬度』坐标来进行定位，而经度为零的地点在英国『格林威治』这个城市所在纵剖面上，(注：所谓的纵剖面就是由南极切到北

极的直线，而横切面就是与赤道平行的切线)，如下图所示：



因为绕地球一圈是 360 度角，这 360 度角共分为 24 个时区，当然一个时区就是 15 度角啦！又由于是以格林威治时间为标准时间(Greenwich Mean Time, GMT 时间)，加上地球自转的关系，因此，在格林威治以东的区域时间是比较快的(+小时)，而以西的地方当然就是较慢啰！以台湾为例，因为台湾所在地为 120 这个东经度，又因为台湾在格林威治的东方 (废话！因为是东经嘛！^_^)，因此，台湾本地时间 (local time) 会比 GMT 时间快 8 小时 (GMT + 8)。当格林威治时间为零点，台湾就已经是早上八点了！底下列出各个时区的名称与所在经度，以及与 GMT 时间的时差：

标准时区	经度	时差
GMT , Greenwich Mean Time	0 W/E	标准时间
CET , Central European	15 E	+1 东一区
EET , Eastern European	30 E	+2 东二区
BT , Baghdad	45 E	+3 东三区
USSR, Zone 3	60 E	+4 东四区
USSR, Zone 4	75 E	+5 东五区
Indian, First	82.3E	+5.5 东五半区
USSR, Zone 5	90 E	+6 东六区
SST , South Sumatra	105 E	+7 东七区
JT , Java	112 E	+7.5 东七半区
CCT , China Coast (台湾所在地)	120 E	+8 东八区
JST , Japan	135 E	+9 东九区
SAST, South Australia	142 E	+9.5 东九半区
GST , Guam	150 E	+10 东十区
NZT , New Zealand	180 E	+12 东十二区

Int'l Date Line	180 E/W	国际日期变更线
BST , Bering	165 W	-11 西十一区
SHST, Alaska/Hawaiian	150 W	-10 西十区
YST , Yukon	135 W	-9 西九区
PST , Pacific	120 W	-8 西八区
MST , Mountain	105 W	-7 西七区
CST , Central	90 W	-6 西六区
EST , Eastern	75 W	-5 西五区
AST , Atlantic	60 W	-4 西四区
Brazil, Zone 2	45 W	-3 西三区
AT , Azores	30 W	-2 西二区
WAT , West Africa	15 W	-1 西一区

所以啰！台湾时间是 GMT + 8 就很容易推算出来了吧！要特别留意的是，很多朋友在安装 Linux 的时候，总是会发现目前的时间慢或者快了 8 小时，不要怀疑，绝对与时区有关！赶紧给他查一下如何调整时区吧！ ^_^。另外，在上表中有个比较有趣的时区，那就是在太平洋上面的国际日期变更线了！我们刚刚说，在格林威治的东边时间会较快，而在西边时间会较慢，但是两边各走了 180 度之后，就会碰头啊！那不就刚好差了 24 小时吗？！没错啦！所以才订定为『国际日期变更线』啊！国际日期变更线刚好在太平洋上面，因此，如果您有坐飞机到美国的经验，应该会发现，咦！怎么出发的时间是星期六下午，坐了 13 个小时的飞机到了美国还是星期六？！因为刚好通过了国际日期变更线，日期减少了一天喔！如果反过来，由美国到台湾，日期就会多加一天喔！ ^_^

什么是夏季节约时间(daylight savings)？

除了时区的概念先建立起来之后，现在再来谈一谈，那么什么是『夏季节约时间』？既然是『夏季节约时间』当然主要是与夏天有关啦！因为地球在运行的时候是呈现一个倾斜角在绕太阳运转的，所以才有春夏秋冬(这个大家应该都知道啦！)，在夏天的时候，白天的时间会比较长，所以，为了节约用电，因此在夏天的时候，某些地区会将他们的时间定早一小时，也就是说，原本时区是 8 点好了，但是因为夏天太阳比较早出现，因此把时间向前挪，在 8 点的时候，订定为该天的 7 点～如此一来，我们就可以利用阳光照明，省去了花费电力的时间，因此才会称之为夏季节约时间！

因为台湾实在是太小了，并没有横跨两个时区，因此，夏季节约时间对我们来说，虽然还是有帮助啦！不过，似乎没有特别推行的样子说～

Coordinated Universal Time (UTC) 与系统时间的误差

了解了一些时区的概念之后，这里要谈的是『什么是正确的时间！』。在计算时间的时候，最准

确的计算应该是使用『原子震荡周期』所计算的物理时钟了(Atomic Clock, 也被称为原子钟)，这也被定义为标准时间(International Atomic Time)。而我们常常看见的 UTC 也就是 Coordinated Universal Time (协和标准时间)就是利用这种 Atomic Clock 为基准所定义出来的正确时间。例如 1999 年在美国启用的原子钟 NIST F-1，他所产生的时间误差每两千年才差一秒钟！真的是很准呐！这个 UTC 标准时间是以 GMT 这个时区为主的喔！所以本地时间与 UTC 时间的时差就是本地时间与 GMT 时间的时差就是了！

事实上，在我们的身边就有很多的原子钟！例如石英表，还有计算机主机上面的 BIOS 内部就含有一个原子钟在纪录与计算时间的进行呐！不过，由于原子钟主要是利用计算芯片 (crystal) 的原子震荡周期去计时的，这是因为每种芯片都有自己的独特的震荡周期之故。然而因为这种芯片的震荡周期在不同的芯片之间多多少少都会有点差异性，甚至同一批芯片也可能会或多或少有些许的差异(就连温度也可能造成这样的误差呢！)，因此，也就造成了 BIOS 的时间会三不五时的给他快了几秒或者慢了几秒。

或许您会认为，BIOS 定时器每天快个五秒也没有什么了不起的，不过如果您再仔细的算一算，会发现，一天快五秒，那么一个月快 2.5 分钟，一年就快了 75 分钟了！所以说，呵呵！时间差是真的会存在的！那么如果您的计算机真的有这样的情况，那要怎么来重新校正时间呢？！呵呵！那就需要『网络校时』(Network Time Protocol, NTP)的功能了！底下我们就谈一谈那个 NTP 的 daemon 吧！

NTP 是什么？

如同前面说的，计算机主机主要是以 BIOS 内部的时间为主要的时间依据，而偏偏这个时间可能因为 BIOS 内部芯片本身的问题，而导致 BIOS 时间与标准时间 (UTC) 有一点点的差异存在！所以，为了避免主机时间因为长期运作下所导致的时间偏差，进行时间同步 (synchronize) 的工作就显的很重要了！

那么怎么让时间同步化呢？想一想，如果我们选择几部主要主机 (Primary server) 调校时间，让这些 Primary Servers 的时间同步之后，再开放网络服务来让 Client 端联机，并且提供 Client 端调整 Client 自己的时间，不就可以达到全部的计算机时间同步化的运作了吗？！那么什么协议可以达到这样的功能呢？！那就是 Network Time Protocol ，另外还有 Digital Time Synchronization Protocol (DTSS) 也可以达到相同的功能！

不过，到底 NTP 这个 daemon 是如何让 Server 与 Client 同步他们的时间呢？！

1. 首先，主机当然需要启动这个 daemon ，之后，
2. Client 会向 NTP Server 发送出调校时间的 message ，
3. 然后 NTP Server 会送出目前的标准时间给 Client ，
4. Client 接收了来自 Server 的时间后，会据以调整自己的时间，就达成了网络校时咯！

不过，在上面的步骤中您有没有想到一件事啊，那就是如果 Client 到 Server 的讯息传送时间过长怎么办？！举例来说，我在台湾以 ADSL 的 PC 主机，联机到美国的 NTP Server 主机进行

时间同步化要求，而美国 NTP Server 收到我的要求之后，就发送当时的正确时间给我，不过，由美国将数据传送回我的 PC 时，时间可能已经延迟了 10 秒钟去了！这样一来，我的 PC 校正的时间是 10 秒钟前的标准时间喔！此外，如果美国那么 NTP 主机有太多的人喜欢上去进行网络校时了，所以 loading (负荷) 太重啦！导致讯息的回传又延迟的更为严重！那怎么办？！

为了这些延迟的问题，有一些 program 已经开发了自动计算时间传送过程的误差，以更准确的校准自己的时间！当然啦，在 daemon 的部分，也同时以 server/client 及 master/slave 的架构来提供使用者进行网络校时的动作!所谓的 master/slave 就有点类似 DNS 的系统咯!举例来说，台湾的标准时间主机去国际标准时间的主机校时，然后各大专院校再到台湾的标准时间校时，然后我们再到各大专院校的标准时间校时!这样一来，那几部国际标准时间主机 (Time server) 的 loading 就不至于太大，而我们也可以很快速的达到正确的网络校时的目的呢!台湾常见的 Time Server 为：

time.stdtime.gov.tw
clock.stdtime.gov.tw
freq_f.stdtime.gov.tw
tick.stdtime.gov.tw
time.chttl.com.tw

至于 ntp 这个 daemon 是以 port 123 为连结的埠口 (使用 UDP 封包)，所以我们要利用 Time server 来进行时间的同步更新时，就得要使用 NTP 套件提供的 ntpdate 来进行 port 123 的联机喔！关于网络校时更多的说明，可以到 NTP 的官方网站上察看喔！

http://www.ntp.org

套件安装：

使用 RPM 安装

一般来说， NTP Server 在各个 distribution 的功能差异应该不很大啦!所以比较建议使用 RPM 的方式来进行安装！您可以拿出 Linux 的原版光盘，mount 上之后，搜寻以 ntp 为开头的套件档名，然后给他安装上去，就可以了！不过，需要特别留意的是，当您安装好了 NTP 之后，系统会自动的将 ntp 启动喔！所以，如果您只是想利用 NTP 套件里面的 Client 功能，那么最好还是将 ntp 这个 daemon 关闭吧！

```
[root@test root]# chkconfig --level 2345 ntpd off
[root@test root]# /etc/rc.d/init.d/ntpd stop
```

如此一来，您的 ntp 套件已经安装完毕，可以使用 ntp 的 client 功能去联机 Time server 进行网络校时了！至于 NTP Server 则需要继续的进行设定呢！

使用 Tarball 安装

使用 Tarball 来安装 NTP 其实也是不困难的！简单的很！请先到 NTP 的官方网站下载最新的 NTP 套件：http://www.ntp.org/downloads.html，这里我以 ntp-4.1.2.tar.gz 这个版本为范例，假设您下载的套件放置在 /root 底下，那么可以这样做：

```
0. 解压缩，并阅读一下 ntp 底下的 README 与 INSTALL：
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/ntp-4.1.2.tar.gz
[root@test src]# cd ntp-4.1.2
[root@test ntp-4.1.2]# vi INSTALL (vi README)

1. 开始设定参数、编译与安装：
[root@test ntp-4.1.2]# ./configure --help | more #可以察看一下可用的参数!
[root@test ntp-4.1.2]# ./configure --prefix=/usr/local/ntp  \
>  --enable-all-clocks --enable-parse-clocks
[root@test ntp-4.1.2]# make clean ; make
[root@test ntp-4.1.2]# make check  #确定一下，是否有问题！？
[root@test ntp-4.1.2]# make install # 将数据给他安装在 /usr/local/ntp 底下
```

设定完成之后，您就有 ntp 可以使用了！不过，无论怎么说，使用 RPM 来安装 NTP 还是比较简单一些啦！ ^_^

Server 端的设定

好了，假设您已经使用 RPM 来安装了 NTP 这个套件，那么我们就可以来谈一谈怎么设定 NTP 主机啦！

NTP 的套件结构

NTP 套件的结果主要分为两部份，一个是 NTP Server 的部分，一个则是 NTP Client 的部分，在这个章节里面，我们不谈比较困难的设定，主要介绍较为简易的设定而已喔！所以，您需要注意到的档案与指令有底下这几个：

- 与 NTP 及时区有关的几个设定档：
  - /etc/ntp.conf ：这个是 NTP daemon 的主要设定档，依据不同的版本放置的目录可能会不同，不过档名都是一样的！使用 locate ntp.conf 搜寻一下您的系统有没有这个档案吧！这也是 NTP 唯一的一个设定档案！
  - /usr/share/zoneinfo：这是个目录，这个目录是 Linux 本身提供的，而不是 NTP 所提供的。在这个目录下的档案其实是规定了各主要时区的时间设定档案，例如台湾地区的时区设定档案在 /usr/share/zoneinfo/Asia/Taipei 就是了！这个目录里面的档案与底下要谈的两个档案(clock 与 localtime)是有关系

的喔！

- /etc/sysconfig/clock：这个档案其实也不包含在 NTP 的 daemon 当中，因为这个是 linux 的主要时区设定档案啊！每次开机后，Linux 会自动的读取这个档案来设定自己系统所预设要显示的时间说！举个例子来说，在我们台湾地区的本地时间设定中，这个档案内应该会出现一行『ZONE="Asia/Taipei"』的字样，这表示我们的时间设定档案『要取用 /usr/share/zoneinfoe/Asia/Taipei 那个档案』的意思！

- /etc/localtime：这个档案就是『本地端的时间设定档』啦！刚刚那个 clock 档案里面规定了使用的时间设定档 (ZONE) 为 /usr/share/zoneinfo/Asia/Taipei ，所以说，这就是本地端的时间了，此时，Linux 系统就会将 Taipei 那个档案复制一份成为 /etc/localtime ，所以未来我们的时间显示就会以 Taipei 那个时间设定档案为准。好了，如果现在我这部主机搬到日本东京去了，那么我应该如何调整时间呢？其实什么调整都不需要，因为我们的 localtime 主要是分析与 UTC 时间的时差来显示的格式，所以，您只要将 /etc/sysconfig/clock 里面的 ZONE 设定成为 Asia/Tokyo 并且将 /usr/share/zoneinfo/Asia/Tokyo 复制成为 /etc/localtime ，呵呵！什么设定都不需要更动，就能显示时间为日本东京的时间了！这样是否能够了解？

- 与 NTP 及时间有关的执行档：

  - /bin/date：这个是 Linux 系统上面常见的日期与时间输出指令，用途很广喔！除了输出时间外，也可以修改时间。

  - /sbin/hwclock：这是一个 root 才能执行的指令，因为 Linux 系统上面 BIOS 时间与 Linux 系统时间是分开的，所以使用 date 这个指令调整了时间之后，还需要使用 hwclock 才能将修改过后的时间写入 BIOS 当中！

  - /usr/sbin/ntpd：这就是 NTP 的主要 daemon 档案啦！得要启动他才能提供 NTP 服务。注意，这个指令预设会参考 /etc/ntp.conf 里面的设定喔！

  - /usr/sbin/ntpdate：这个就是 Client 端用来连接 NTP Server 的主要执行档啰！如果您没有要启用 NTP 而仅想要使用 NTP Client 功能的话，那么只会用到这个指令而已啦！

  - /usr/sbin/ntptrace：这个指令可以用来追踪某部时间服务器的时间对应关系，这也是个很有用的指令喔！底下我们会介绍如何使用这支程序！

主机的规划技巧建议

因为 NTP daemon 并没有花费什么硬盘空间，所以主机的规划上面就没有太多的考虑了！

编辑主要设定档 /etc/ntp.conf

在 NTP Server 的设定上面，其实最好不要对 Internet 无限制的开放，尽量仅提供您自己内部的 Client 端联机进行网络校时就好。此外， NTP Server 总也是需要网络上面较为准确的主机来自行更新自己的时间啊，所以在我们的 NTP Server 上面也要找一部最靠近自己的 Time Server 来进行自我校正喔！事实上，就如同前面的说明， NTP 这个服务也是 Server/Client 的一种模式（当然也提供 Peer/Peer ，不过我们这里主要讨论 Server/Client 的架构 ），在 Internet 上面提供了多部主要的 (Primary) 时间服务器，如下的网页连结所示。不过，虽然 Primary Time Servers 不少，然而 Client 数量更是大的惊人，为了不让 Primary 的时间服务器负载太大，所以就有 Secondary (次要的) 时间服务器的出现了！这个 Secondary 主要就是利用 Primary 主机进行时间调校后，再提供 Internet 上面的 Client 进行校时。

- 主要时间服务器：http://www.eecis.udel.edu/~mills/ntp/clock1a.html
- 次要时间服务器：http://www.eecis.udel.edu/~mills/ntp/clock2a.html

如果想要在台湾地区进行网络校时，那么 time.stdtime.gov.tw 这个国家单位的第二层主机(stratum-2) 应该是比较合适的！一般来说，我们在进行 NTP 主机的设定时，都会先选择数部上层的 Time Server 来做为我们这一部 NTP Server 的校正之用，选择多部的原因是因为可以避免因为某部时间服务器突然挂点时，其它主机仍然可以提供我们的 NTP 主机来自我更新啊！然后我们的 NTP Server 才提供给自己的 Client 端更新时间。如此一来，国家单位的 time.stdtime.gov.tw 负载才不会太大，而我们的 Client 也可以很快速的达到校时的动作！

这里还是需要注意一下，台湾地区的主要 (或称为第一层 stratum-1) 时间服务器 IP 为：

- 210.59.157.40
- 210.59.157.41
- 210.59.157.151

而第二层的主机有很多，例如：

- 210.59.157.10
  210.59.157.30
  202.39.157.155

基于上面的说明，我的 NTP 服务器主要的设定项目是这样的：

- 以上面提到的六部时间服务器作为我的 NTP server 的上层主机;
- 不对 Internet 提供公开的服务，仅针对内部网域 192.168.0.0/24 提供服务;
- 亦对网络上 192.168.100.20 这个 IP 提供服务;
- 内部私有网络的网络校时不需要认证机制;

这样的设定真的是很简单喔！我们就来设定一下吧！

```
[root@test root]# vi /etc/ntp.conf
# 1. 关于权限设定部分
#    权限的设定主要以 restrict 这个参数来设定，主要的语法为:
#
#    restrict IP mask netmask_IP parameter
#
#    其中 IP 可以是软件地址，也可以是 default ，default 就类似 0.0.0.0 咯!
#    至于 paramter 则有:
#      ignore  : 关闭所有的 NTP 联机服务
#      nomodify: 表示 Client 端不能更改 Server 端的时间参数，不过，
#                Client 端仍然可以透过 Server 端来进行网络校时。
#      notrust : 该 Client 除非通过认证，否则该 Client 来源将被视为不信任网域
#      noquery : 不提供 Client 端的时间查询
#    如果 paramter 完全没有设定，那就表示该 IP (或网域) 『没有任何限制！』
#
#    在我们这个例子当中，因为拒绝所有，仅开放 192.168.0.0/24 ,
#    并且让 127.0.0.1 以及本机 IP 192.168.0.2 可以不受限制，所以:
restrict default ignore  # 关闭所有的 NTP 要求封包
restrict 127.0.0.1       # 开启内部递归网络接口 lo
restrict 192.168.0.2    # 主机本身的 IP 也同时开启!
restrict 192.168.100.20 mask 255.255.255.255 nomodify
# 针对另一个 IP 开放让他可以更新时间!
restrict 192.168.0.0 mask 255.255.255.0 nomodify
# 在网域里面的 client 可以进行网络校时，但不会影响 Server !

# 2. 上层主机的设定
#    上层主机我们选择 time.stdtime.gov.tw ，要设定上层主机主要以 server
#    这个参数来设定，语法为:
#
#    server [IP|FQDN] [prefer]
#
#    Server 后面接的就是我们上层 Time Server 啰！而如果 Server 参数
#    后面加上 perfer 的话，那表示我们的 NTP 主机主要以该部主机来作为
#    时间校正的对应。另外，为了解决更新时间封包的传送延迟动作,
#    所以可以使用 driftfile 来规定我们的主机
#    在与 Time Server 沟通时所花费的时间，可以记录在 driftfile
```

```
#    后面接的档案内，例如下面的范例中，我们的 NTP server 与
#    time.stdtime.gov.tw 联机时所花费的时间会记录在 /etc/ntp/drift 档案内
# 先输入第二层主机的 IP
server 210.59.157.10   prefer
server 210.59.157.30   prefer
server 202.39.157.155  prefer
# 第一层的主机就列为参考用!
server 210.59.157.40
server 210.59.157.41
server 202.39.157.151
# 当然要让 Server 可以进入我们的 NTP 主机啦! 权限要开放啊!
restrict 210.59.157.10
restrict 210.59.157.30
restrict 202.39.157.155
restrict 210.59.157.40
restrict 210.59.157.41
restrict 202.39.157.151
driftfile /etc/ntp/drift
```

在上面的设定当中，最有趣的应该要算 driftfile 那个咚咚了！因为我们的 NTP Server 本身的时间计算是依据 BIOS 的芯片震荡周期频率来计算的，但是这个数值与上层 Time Server 不见得会一致啊！所我 NTP 这个 daemon 会自动的去计算我们自己主机的频率与上层 Time server 的频率，并且将两个频率的误差记录下来，记录下来的档案就是在 driftfile 后面接的完整档名当中了！我们这里是以预设的档案 /etc/ntp/drift 来设定，您也可以自行设定其它的档名，不过请注意：

- driftfile 后面接的档案需要使用完整路径文件名；
- 该档案不能是连结档；
- 该档案需要设定成 ntpd 这个 daemon 可以写入的权限。

driftfile 后面接的档案会被 ntpd 自动更新，所以他的权限一定要能够让 ntpd 写入才行。在 Red Hat 9 预设的 NTP 服务器中，使用的 ntpd 的 owner 是 ntp ，所以 /etc/ntp/drift 需要设定成 ntp 这个 user 可以写入喔！至于 owner 怎么会是 ntp 呢？请查阅 /etc/sysconfig/ntpd 就可以知道啦！而 /etc/ntp/drift 的内容则是仅有一行，里面的数据是具有小数点的浮点数字，单位则是百万分之一(ppm)。

事实上 ntp.conf 里头还有很多很有趣的设定，例如认证的机制、登入的 Client 观察以及其它的相关设定等等，不过这里我们就不多加介绍了！

NTP 的启动与观察

在设定好了 ntp.conf 之后，就可以来启动 NTP 这个 Time Server 了！我们可以这样做喔：

```
[root@test root]# /etc/rc.d/init.d/ntpd start
[root@test root]# netstat -unl | grep 123
udp        0      0 192.168.0.2:123         0.0.0.0:*
udp        0      0 127.0.0.1:123           0.0.0.0:*
udp        0      0 0.0.0.0:123             0.0.0.0:*
# 请注意喔，NTP 使用的是 UDP 的封包！而且 port number 为 123，
# 此外，我有两个接口，以及对外提供服务，所以自然就会有三个！

[root@test root]# ntptrace 192.168.0.2
192.168.0.2: stratum 3, offset 0.000056, synch distance 0.65865
210.59.157.10: stratum 2, offset -0.228265, synch distance 0.22488
ntp0.usno.navy.mil: stratum 1, offset -0.250685, synch distance 0.00038,

# 我们也可以利用 ntptrace 来追踪一下到底我们的主机有没有正确的先经过
# 上层 Time Server 的校时了呢？如上所示，我们的主机 192.168.0.2
# 是第三层的时间服务器 (stratum 3)，与目前本机的时间误差(offset)
# 以及若要同步更新时(synch distance)的时间损耗，同时，也会将这个第三层
# 对哪一部第二层主机进行校时也列出来！有时候您会发现这样的错误讯息：
192.168.0.2: stratum 16, offset 0.000048, synch distance 0.00087
0.0.0.0:        *Not Synchronized*
# 这表示我们的主机尚未与 Internet 的上层 Time Server 进行校时，
# 最可能发生这样的错误在 ntp.conf 里面的 restrict 设定了！
# 此外，当启动 ntpd 后，您至少需要等待 5 分钟左右，这段时间
# 我们的 NTP Server 会不断的与上层时间服务器联系，如果尚未联系成功，
# 那么我们的 NTP 主机就会暂时无法让 Client 端来进行更新喔！
# 所以如果未能更新，不要太紧张，先等待一阵子再说吧！
```

如果 ntptrace 可以成功的话，那就表示您的主机 OK 啦！

Client 端的设定：

好了，再来就是要在 Client 端来向 Time Server 要求网络校时啦！不过，我们先谈一谈如何手动修正时间吧！

如何调整 Linux 系统的时区与手动设定时间 (date MMDDhhmmYYYY)

我们在前面说过， Linux 的时区档案放置在 /etc/localtime ，这是一个时间格式的档案，而不是 ASCII 类型的档案喔！(file /etc/localtime 可以看出)，至于所有的 Time Zone 则放置在 /usr/share/zoneinfo 这个目录下。请注意：

- 当 /etc/localtime 存在时，系统的时区以该档案代表的时区来显示、
- 当 /etc/localtime 不存在时，系统的时区主要以 GMT (或 UTC) 为准；

所以，如果您想要变更您 Linux 系统的时区，那么只要在 /usr/share/zoneinfo 里面找到您需要的时区档案，然后将他复制一份成为 /etc/localtime 就可以顺利的更新时区设定了！另外，同时建议修正一下 /etc/sysconfig/clock 这个档案里面的 ZONE 设定值！以我们台湾的 Time zone 为例，在 /etc/sysconfig/clock 这个档案当中应该是『ZONE="Asia/Taipei"』这就表示我们的时区档案为 /usr/share/zoneinfo/Asia/Taipei 这个档案啰!请对应着修改成您所想要的时区吧！

好了，时区修正完毕了，那么时间呢？！呵呵！目前 Linux 系统上面有两个时间喔，一个是 Linux 系统，另一个则是 BIOS 时间(真正的硬件记录的时间)！我们可以使用 date 这个指令来手动修正目前主机的时间，不过， date 这个指令仅修正 Linux 时间而已，我们还需要以 hwclock 这个指令来将 BIOS 时间也更新才行！

```
[root@test root]# date MMDDhhmmYYYY
MM：月份
DD：日期
hh：小时
mm：分钟
YYYY：公元年
[root@test root]# date 082110002003
Thu Aug 21 10:00:00 CST 2003
# 时间立刻就修正了！

[root@test root]# hwclock [-rw]
-r：检视目前的 BIOS 时间
-w：将目前 Linux 的时间写入 BIOS 当中！
[root@test root]# date ; hwclock -r
Thu Aug 21 10:01:46 CST 2003
Thu 21 Aug 2003 09:57:52 AM CST  0.647923 seconds
# 你可以看到， date 与 hwclock -r 所显示的时间是『不一致的』！
# 这就是因为 Linux 时间与 BIOS 时间不一致所导致的一个问题！
# 我们需要以 hwclock -w 来将 Linux 时间写入 BIOS 喔！
[root@test root]# hwclock -w
[root@test root]# date ; hwclock -r
Thu Aug 21 10:03:42 CST 2003
Thu 21 Aug 2003 10:03:43 AM CST  0.113323 seconds
# 呵呵！这样时间就一致啦！
```

这样可以了解了吗？!没错,当我们进行完 Linux 时间的校时后,还需要以 hwclock 来更新 BIOS 的时间，因为每次开机的时候，系统会重新由 BIOS 将时间读出来，所以， BIOS 才是重要的时间依据呐!

如何在 Linux 系统自动网络校时?

在 Linux 上面进行网络校时简单的很，直接以 ntpdate 这个指令来执行即可！如下所示：

```
[root@test root]# ntpdate 192.168.0.2
# 那个 192.168.0.2 是我们刚刚建立的 NTP Server ，您也可以选择
# time.stdtime.gov.tw 这部主机来校时喔!
21 Aug 10:05:29 ntpdate[23420]: step time server 192.168.0.2 offset -236.117047 sec
[root@test root]# hwclock -w
# 这样时间就修正了！若要每日进行时间校正，可以写入 cron

[root@test root]# vi /etc/crontab
# 加入这一行：
10 5 * * * root /usr/sbin/ntpdate 192.168.0.2 ; /sbin/hwclock -w
```

使用 cron 之后，每天 5:10 Linux 系统就会自动的进行网络校时啰！相当的简易吧!

如何在 Windows 系统上面进行网络校时?

在 Windows 上面进行网络校时也很简单，目前已经有很多热心人士写好了在 Windows 上面的网络校时软件了！例如全中文接口的 ntpclock1_21.exe ，您可以在以下的网站下载：

- http://www.stdtime.gov.tw/ntp/index.htm
- http://www.stdtime.gov.tw/ntp_client/menu/NTPClockOperatingManualRelease1.21.html

上面同时列出使用说明，请自行参考喔！当然，您也可以在鸟哥的网站下载：

- http://linux.vbird.org/download/#ntp_win

安全相关方面

NTP 服务器在安全的相关性方面，其实刚刚我们在 /etc/ntp.conf 里面的 restrict 参数中就已经设定了 NTP 这个 daemon 的服务限制范围了！不过，在防火墙 iptables 的部分，还是需要启用的啦！所以，在您的 iptables 规则的脚本 (scripts) 当中，需要加入这一段 (我是以开放 192.168.0.0/24 这个网域作为范例的！)

```
/sbin/iptables -A INPUT -p UDP -i eth0 -s 192.168.0.0/24  \
> --dport 123 -j ACCEPT
```

若还要开放其它的网段或者主机，请自行修改您的防火墙机制咯！

本章与 LPI 的关系

在 LPI 网站 http://www.lpi.org 里面提到的，在 LPI 102 里面 NTP 其实考的还不少！Topic 1.111.6 Maintain system time 里面说到，应试者应该要了解 BIOS 时间与 UTC 时间的意义，同时需要知道怎么设定 timezone ，而时间差所使用的档案 driftfile 也需要了解呐！可能会考的内容含有：

- date
- hwclock
- ntpd
- ntpdate
- /usr/share/zoneinfo
- /etc/localtime
- /etc/ntp.conf
- /etc/ntp.drift (新版已经改至 /etc/ntp/drift 了)

参考资源：

- NTP 的官方网站：http://www.ntp.org
- 成大计中：http://turtle.ee.ncku.edu.tw/~antony/xntpd/
- 台湾地区时间服务器：http://www.stdtime.gov.tw/ntp/index.htm
- NTP 网站：http://www.eecis.udel.edu/~mills/ntp/html/ntpd.html

课后练习

- 什么是 GMT (格林威治) 时间与 UTC 时间？
- Linux 系统的所有时区档案放置哪一个目录底下？
- 我的 Linux 主机本来放置在日本东京，现在想将他拿到台湾来运作，不过因为日本与台湾有一个小时的时差，所以我的时间应该需要经过调整才行。不过，因为我的 BIOS Time 主要是依据 UTC 时间来设定的，所以似乎只要更动时区参数即可。请问我该如何设定时区，好让我的 Linux 主机能够显示正确的时间？
- 目前 Linux 系统上面的时间服务器主要是以 NTP 为主，请问这个 daemon 的主要设定档放在哪里，而该设定档中，针对上层 time server 的设定参数为何？而那个 driftfile 参数是干嘛用的？
- 请问 ntptrace 的功能为何？
- 我以 date 更新了我 Linux 上面的时间后，该如何将时间数据写入 BIOS 内？
- 在 Linux 上面如何进行网络校时？

简易 APT/YUM 服务器设定

最近更新日期：2004/06/23

用不惯 Tarball 安装套件却又担心 RPM 的属性相依问题吗？如果有一种套件管理工具可以克服 RPM 属性相依的方法该有多好！有没有这种工具？呵呵！有的，那就是 APT 与 YUM 这两个服务器了。APT (Advanced Package Tool) 是由 debian 这个 distribution 所发展的一个套件管理工具，其目的在克服 RPM 套件的属性相依问题，让使用者可以透过 APT 的分析直接安装/升级/删除相关联的套件喔。另一个很好用的就是 YUM (Yellow dog Updater, Modified) 这个咚咚，他是由 Duke University 所发起的计划，目的则与 APT 相似，都是在克服 RPM 的属性相依问题，方便使用者进行套件的安装、升级等等工作。由于 APT/YUM 这一类的服务器在『系统升级/管理』上面的功能发挥的很好，所以目前很多的 distributions 都把这两个服务器作为预设的服务喔。在这个章节当中，我们要介绍如何在您的 Linux 服务器上面建置一个 ATP 或 YUM 服务器，并且提供更新的 RPM 套件给 Client 端来使用！

前言：

如果您曾经自行安装过某些套件的话，那么您或许会觉得：『RPM 是比 Tarball 好安装没有错啦，但是每次为了解决套件之间的属性相依问题，真的是很烦，尤其是 RPM 档案在不同的操作系统版本之间也无法兼容！』是的！没错！为了要解决这个套件之间的属性相依问题，又要保留 RPM 套件的易安装与查询的特性，所以就有一些套件管理方法出来啦！比较有名的就是 APT 与 YUM 这两个咚咚啦。(注：当然，Mandrake 的 urpmi 也是很棒的工具。)

甚么是 APT/YUM 呢？他们如何运作？

众所皆知的，RPM 是目前 Linux 世界里面用的最多的套件安装方式。不过，由于 RPM 所管理的套件在安装的时候必须要考虑到不同套件之间的相依性，这在系统管理员处理系统的升级/更新上面是很讨厌的！为了克服这个问题，所以有 APT/YUM 之类的计划出来。APT (Advanced Package Tool) 最早是由 debian 这个 distribution 所发展出来的，而 YUM (Yellow dog Updater, Modified) 则是由 Duke University 所发起的计划之一，这两者的目的都是『为了要解决安装 RPM 套件时的属性相依问题！』，而不是额外再建立一个套件安装模式喔。首先，我们谈一谈为甚么 RPM 套件会有属性相依的问题？又，甚么是属性相依的问题啊？

- 属性相依的问题大多来自于函式库的引用，举例来说，我们前面 SSH 与 Telnet 服务器章节中提到的 SSH 服务器需要使用到 SSL 这个套件的加密机制，所以自然就需要 SSL 的函式库，这个时候，如果您没有安装 SSL 的话，那么 SSH 就不会让您安装了！这就是属性相依的问题啦。也就是说，当我们要安装 A 套件，结果 RPM 套件管理员告诉你还需要 B 套件，而安装 B 套件时，却又发现还缺少 C 套件～真是环环相扣啊！讨厌死了～

- 要知道某个套件的最低要求是哪些套件时，可以使用 rpm -qR packagename ，至于某个套件提供哪些档案则可以使用 rpm -q --provides packagename 。这些信息都纪录在 RPM 套件里面。

如果对于 RPM 还有疑问，请参考基础学习篇的 RPM 与 SRPM 那个章节啰。好了，既然每一个 RPM 套件的标头 (header)里面都会纪录该套件的属性相依关系，那么如果我们可以将该标头的内容纪录下来并且进行分析，不就可以得知每个套件在安装之前需要额外安装哪些基础套件吗？也就是说，我们在服务器上面先以分析工具将所有的 RPM 档案进行分析，然后将该分析纪录下来，只要在进行安装或升级时先查询该纪录的档案，就可以知道所有相关联的套件档案！没错！是这样。他的整个运作流程有点像这样：

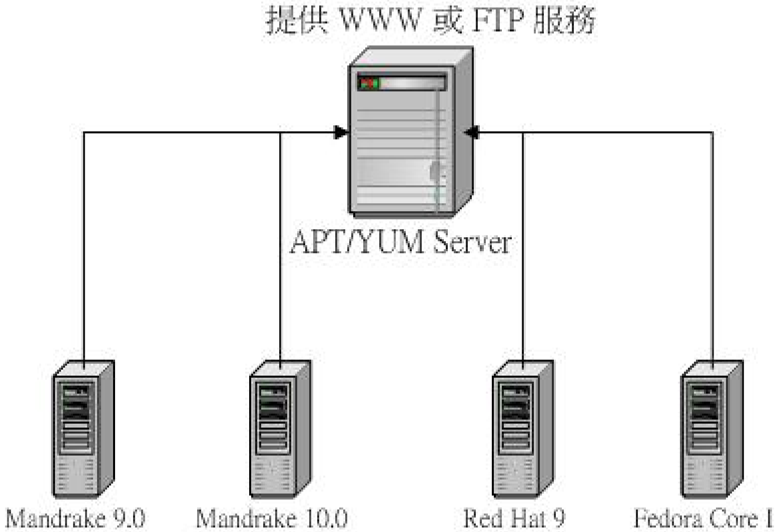
服务器端：

3. 首先，在 APT/YUM 服务器上面放置了所有的 RPM 套件(这包括来自原版光盘与供货商发行的升级套件)；
4. 然后以相关的功能去分析各个 RPM 档案的相依性，这些纪录可以解决所谓的属性相依问题，然后将这些数据记录成档案存放在服务器的某特定目录内；

客户端：

5. Client 端如果需要安装/升级/删除某个程序时，会先下载服务器上面记载的属性相依档案(利用的协议则是 WWW 或者是 FTP)；
6. 经由比对服务器端传来的纪录数据进行分析，然后取得所有相关的套件，一次全部下载下来进行升级安装。

如此一来则克服了属性相依的问题啰！是的！就这么简单啊！整个图示如下：



图一、APT/YUM 服务器的架构。

您的 APT/YUM 服务器上面可以拥有多个版本的 Linux distributions 的 RPM 套件，并且需要提供 WWW/FTP 等服务，而 Client 端则是藉由主机的 WWW/FTP 等协议来进行 RPM 档案的取得喔。

是否需要架设 APT/YUM 服务器

APT/YUM 的功能也只是在管理 RPM 套件而已，只是他比单纯的 RPM 指令要好的地方在于他克服了属性相依的问题，所以客户端可以很方便的进行安装/升级与移除的动作。那么是否意味着我就得要架设 APT 或 YUM 服务器呢？

这可不一定呐。如果您只有一部主机，而且上面的网络服务很少，并且也没有提供甚么重要的服务，那么架设 APT/YUM 服务器就没有这么需要了。为甚么呢？因为很少用到 APT 的机制啊，而且直接找网络上面有的 APT/YUM 服务器即可，无须架设啦！要注意的是，架设 APT/YUM 服务器时，您必须要有够用的硬盘空间、够用的频宽以及提供至少 WWW 或者是 FTP 的网络服务呢！这对只有一部主机的您来说，真的不必要费心思去管理的啦！但如果您的网络环境是如下的模样，可能就得要架设一部 APT/YUM 服务器比较好啰！

- 您的网络里面有相当多的同样版本的 Linux distribution 系统；
- 您的局域网络对外频宽不高，且内部有多部 Linux 主机系统；
- 您的 Linux distributions 在国内并没有相对应的 APT/YUM 服务器提供服务，而对国外联机的频宽又很低时；

也就是说，如果您拥有多部 Linux 主机，或者是您连接到 APT/YUM 服务器的频宽太低时，就可以考虑架设 APT/YUM 服务器了。以上面图一来说，如果您的连外网络频宽太低，那么架设一部 APT/YUM 服务器后，所有的 Linux 是连接到该部 APT/YUM 服务器进行升级/安装，速度当然比连外要快很多啦！

所以说，如果您是上层的系统管理员(例如学校的网管老师或者县网、区网的网管人员)，为了您的整个网域的 Linux 主机来打算，那么架设 APT/YUM 是蛮需要的，因为真的可以节省很多连外的频宽;相反的，如果您是一般用户，拥有的了不起就是两三部 Linux 主机而已，那么似乎没有架设 APT 主机的必要性呢！因为由上面的说明来看，架设 APT 主机所需要的『硬盘空间』可是不能省的，对于一般的用户来说，架设 APT 主机实在是没有什么必要性说～

架设之前，您所需要启用的服务

就如同前面的说明，您要架设 APT/YUM 时，请先记得在您的 APT/YUM 主机上面启用 WWW 或者是 FTP 服务了才行！因为 APT/YUM 是利用 WWW/FTP 来进行 RPM 档案在 Server 与 Client 之间的传送的！此外，您的 APT/YUM 主机上面最好将最靠近您主机的 Linux 版本更新套件的网站设定为映像站台 (mirror)，如此一来，您就可以让系统自动的更新您的 Linux distributions 所需要的更新档案，而不必手动来更新呢！

此外，由于完整的 APT/YUM 服务器包含了原本的 Linux distributions 的原版光盘内容，所以需要的硬盘空间是很高的！至少需要 3~5 GB 以上，最好能够有 10GB 以上的硬盘空间。

好了，底下我们就来开始安装与设定 APT/YUM 服务器吧！

APT 服务器：

底下开始来谈一谈 APT 服务器吧！

APT 服务器利用的机制

就如同前言说明的， APT 主机可以将已经存在的 RPM 档案进行分析，并且将各个套件的相关系记录下来，以便让使用者依据这个套件的相关性纪录档案来更新与安装他们的 Linux 系统。那么这些档案放在哪里呢？其实因为这些档案被需要让 Client 端可以下载，而且 APT 并不是额外再启用其它的 port ，而是透过 WWW 或者是 FTP 的方式来让 Client 端下载的，所以 APT 的 RPM 档案当然就需要放在可以让 WWW 或者让 FTP 来存取的目录啰！

在很多 Linux distributions 的 WWW 预设主页是放在 /var/www/html 这个目录下的 (例如 Red Hat 9, Fedora Core I/II, Mandrake 等等)，所以，除非您想要自行架设虚拟主机，或者是利用连结档的方式来让 RPM 档案放置的目录可以让 WWW 读取，否则您都应该要将 APT 管理的 RPM 档案放置在 /var/www/html 底下。假设我有两个 Linux 的版本要这部 APT 主机管理时，一版是 Mandrake 9.1 一版是 Red Hat 9 ，那么我可以这样编排我的 RPM 档案放置的目录：

表一、APT 主机相关 RPM 档案放置的目录示意表

```
/var/www/html/apt/redhat9
                        |--RPMS.os
                        |--RPMS.updates
                        |--SRPMS.os (非必备目录)
                       |--SRPMS.updates (非必备目录)
                        `--base
                              |--pkglist.os.bz2
                              |--pkglist.updates.bz2
                              |--release
                              |--release.os
                              `--release.updates
/var/www/html/apt/mdk9.1
                      |--RPMS.os
                      |--RPMS.updates
                      |--SRPMS.os (非必备目录)
                     |--SRPMS.updates (非必备目录)
                      `--base
                            |--pkglist.os.bz2
                            |--pkglist.updates.bz2
                            |--release
                            |--release.os
                            `--release.updates
```

如上所示，我独立出一个名为 apt 的目录来管理我的 APT 档案，另外，因为有两个版本，所以我将 apt 又分为两个目录，因为两个目录的格式一样，所以我以 Red Hat 9 那个 /var/www/html/apt/redhat9 目录来介绍。里面至少会有三个目录才对，分别是 RPMS.os, RPMS.updates, base 这三个目录，其中：

- RPMS.os 的内容为 Red Hat 9 的原本光盘中的 RPM 档案(i386)；
- RPMS.updates 的内容为 Red Hat 公司针对 Red Hat 9 这一版所释出的修补套件；
- base 这个目录里面的数据则是由 APT 服务器所自动产生的，前面我们提到的 APT 会去分析 RPM 档案而将档案信息记录下来，记录下来的咚咚就是放置在这个目录下的啦！

您还会看到其实还有 SRPMS.os 以及 SRPMS.updates 等目录，呵呵！那个就是 Source RPM (SRPM) 档案放置的目录了。因为我们直接可以透过 Binary 来升级，所以我预设不放 SRPM 在我的 APT 服务器里面啰！

OK！所以我们需要怎么来架设我们的 APT 以及利用 APT 来升级呢？

4. 先将所有来自 Linux 原版光盘的 RPM 档案复制到 /full/path/RPMS.os 档案中；
5. 再将来自原 Linux 版本公司释出的 RPM 修补套件由 Internet 下载到 /full/path/RPMS.updates 当中；
6. 利用 APT 的功能进行 /full/path/base 这个目录里面信息的更新；
7. 到 Client 端上面，以 APT 的功能更新 Client 自己的套件信息，使与 APT 主机的 /full/path/base 这个目录里面的套件记录信息同步化；
8. 到 Client 端上面，可以利用 APT 的功能来更新或者是安装套件了！当然，所使用来下载 RPM 档案的协议当然是 WWW 或是 FTP 啰！而且， Client 端分析的是自己的 RPM 信息喔，所以每次进行更新之前，应该将 Client 端的 RPM 信息与 APT Server 端来同步化才行！否则 Server 新增的档案在 Client 端是无法取得信息的喔！

所以啰，其实在 APT 主机设定上面最重要的步骤应该是在于主机硬盘的规划、档案的复制与 updates 档案的随时自动更新(可以利用映像站台的功能)，以及 /full/path/base 目录下的信息更新等等步骤！只要这边没有问题，其它的流程就简单咯！

安装 APT 软件

因为 APT 这套件是挺大的，而且也不容易使用 Tarball 来安装，所以比较建议使用 RPM 的方式直接安装，您可以到底下的网站搜寻一下属于您自己的 Linux distributions 的 APT 版本喔：

- http://rpmfind.net/linux/rpm2html/search.php?query=apt
- http://apt.freshrpms.net/

同时我也将一些常见的版本捉下来了，您也可以在这里下载：

RPM 的部分：

- Red Hat 9：http://linux.vbird.org/download/linux_server/apt/rh9/apt-0.5.5cnc6-fr1.i386.rpm
- Red Hat 8：http://linux.vbird.org/download/linux_server/apt/rh8/apt-0.5.5cnc6-fr0.rh80.1.i386.rpm
- Red Hat 7.3：http://linux.vbird.org/download/linux_server/apt/rh7.3/apt-0.5.5cnc5-fr0.rh73.2.i386.rpm
- Red Hat 7.1：http://linux.vbird.org/download/linux_server/apt/rh7.1/apt-0.5.4cnc9-fr0.1.rh71.i386.rpm
- Mandrake 9.1：http://linux.vbird.org/download/linux_server/apt/mdk9.1/apt-0.5.5cnc3-1mdk.i586.rpm
- Mandrake 9.0：http://linux.vbird.org/download/linux_server/apt/mdk9.0/apt-0.3.19cnc55-2mdk.i586.rpm
- Mandrake 9.0：http://linux.vbird.org/download/linux_server/apt/mdk9.0/apt-devel-0.3.19cnc55-2mdk.i586.rpm

SRPM 的部分(可以使用 Red Hat 9 提供的 SRPM 来进行重新编译)：

- Red Hat 9：http://linux.vbird.org/download/linux_server/apt/rh9/apt-0.5.5cnc6-fr1.src.rpm

直接以『 rpm -ivh package.name 』就能安装了！当然，需要选择适合您的版本呐！如果找不到适合您的版本，那么尝试以上面提供的 SRPM 来重新编译试看看能不能成功吧！『rpm --rebuild SRPM 』或新版的『rpmbuild --rebuild SRPM』(注：SRPM 表示您由上面下载的 apt-0.5.5cnc6-fr1.src.rpm 这个档案的档名)。

APT 服务器的套件结构

APT 服务器里面有很多的档案，说明如下：

- 设定档案：

- /etc/apt/apt.conf：这个并不是 APT Server 的记录文件，而是当我们在指令列模式下达 APT 的指令时 (如底下执行档部分会介绍的 apt-get)，该指令的环境参数。一般来说，使用默认值就可以了！不需要更动他。这个档案的内容当中：

  批注符号为两个斜线(slash)：『//』
  主要至少分为三大群组，分别为 APT(环境参数), Acquire(下载相关参数) 与 RPM(RPM 相关参数)，而每个群组之内又分别具有多个参数，每个参数的设定值最后以分号『;』隔开，例如下面的范例：

```
[root@test root]# vi /etc/apt/apt.conf
APT        //第一个大群组
{
  // Options for apt-get
  Get      //第一个大群组里面的第一个参数
  {
    Download-Only "false";  //第一个参数的项目与该项目之设定值
    Show-Upgraded "true";   //第二个参数的项目与该项目之设定值
  };
};
```

- 除了上面的格式外，也可以利用底下的格式来进行设定：
  主群组::参数::项目 "设定值";
  例如上面的范例中，可以将第一个设定值写成：
  APT::Get::Download-Only "false";

  另外，如果您想要使用 Proxy 来加快您的网络传输时，可以修改里面的内容，因为 Proxy 是在 Acquire 里面的 Http 参数，所以您可以使用如下的设定值 (注：我以成大的 proxy.ncku.edu.tw:3128 为例)：

```
1. 利用原本的设定技巧：
[root@test root]# vi /etc/apt/apt.conf
// 找到底下的参数
Acquire
{
  // 底下加入这些数据：
  Http
  {
    Proxy "http://proxy.ncku.edu.tw:3128";
  };
  Retries "0";
};
```

```
2. 或者您也可以改用底下的参数设定：
[root@test root]# vi /etc/apt/apt.conf
//在最后一行加入，注意，不要被括号 {} 括住了！
Acquire::Http::Proxy "http://proxy.ncku.edu.tw:3128";
```

- 上面两种方法都是行的通的啦！

- /etc/apt/sources.list：这个档案就真的重要了！此档案的作用在于『选择适合您的 APT 主机』啰！所以这个档案与 Client 的关系比较大。内容有点像这样：

```
[root@test root]# vi /etc/apt/sources.list
# Red Hat Linux 9
rpm http://ayo.freshrpms.net redhat/9/i386 os updates freshrpms
#rpm-src http://ayo.freshrpms.net redhat/9/i386 os updates freshrpms

# 里面料的格式为：
# rpm <APT 服务器地址> <相对于服务器的路径> <目录一> <目录二> <目录三> ...
# 以上面的例子来说，事实上 RPM 放置的目录在：
# http://ayo.freshrpms.net/redhat/9/i386/RPMS.os
# http://ayo.freshrpms.net/redhat/9/i386/RPMS.updates
# http://ayo.freshrpms.net/redhat/9/i386/RPMS.freshrpms
# 而至于 rpm-src 则是放置 SRPM 档案的服务器与目录喔！
```

- 如果您是由鸟哥面推荐的 RPM 安装您的 APT 时，这里可以保持预设的路径，不过，如果您知道台湾地区有更快速的映射站台，这里就可以修改成您所找到的 APT 服务器啰。

- 执行档案：

  - apt-get：这个是最主要的执行档了！大部分的时候都是给 Client 端用的，语法如下：

```
[root@test root]# apt-get <options> <更新项目> <套件名称>
参数说明：
options：关于参数有底下几个较常见的：
    -q 不要显示 apt-get 运作时的输出讯息，安静一点比较好吗？！ ^_^
    -y 如果 apt-get 在工作过程中需要使用者响应，这个参数可以直接回答 yes
更新项目：更新的动作有底下几个：
```

```
    update：这个动作很重要，就是我们上面有提到的，Client 端要更新与 APT Server
            套件相关性档案的清单对应表，就得要使用这个项目了！基本上，
            每次进行 apt-get 来下载 APT Server 的档案前，最好都先 apt-get update
    install：安装某个套件，后面接套件名称
    dist-upgrade：自动升级我们系统上面已经安装的所有 RPM 套件喔
    clean：将下载自 APT 主机的的 RPM 档案删除哩！
    remove：移除已经安装在我们系统的某个套件！
范例：
[root@test root]# apt-get update           # 将 RPM 档案相关性清单更新！
[root@test root]# apt-get install tcpdump # 安装 tcpdump 这个套件
[root@test root]# apt-get -y dist-upgrade  # 升级我们系统上面的所有 RPM 套件
[root@test root]# apt-get clean

# 至于每日更新的话，可以写入 /etc/crontab 喔
[root@test root]# vi /etc/crontab
40 5 * * * root apt-get update; apt-get -y dist-upgrade ; apt-get clean
```

- genbasedir：我们在前言的部分一再地提到 APT Server 会分析已经存在的 RPM 档案的属性相关性，并且会将属性的结果放置在 /full/path/base 那个目录内～呵呵！其实该动作很简单的，就是使用这个 genbasedir 即可！

- 相关目录：
  - /var/cache/apt：一些记录档案的地方，例如当使用 apt-get update 之后，这个目录下的 RPM 档案相关系记录文件就会更新了！
    /var/state/apt：这个则是 apt 在工作的时候，一些状态的纪录档案放置的地方！

APT 服务器设定 (以 HTTP 提供服务为例)

在底下的例子当中，我们主要是设定以 HTTP 为 APT Server 的服务提供者，因为是利用预设的 WWW 系统版本，所以首页在 /var/www/html。鸟哥在 /var/www/html 底下建立一个名为 apt 的目录，并在底下提供 Red Hat 9 与 Mandrake 9.1 的 APT 套件服务功能，注意，我预设仅提供 RPM 档案，并不提供 SRPM 的档案喔！相关的目录如前面提到的表一所示(注：当然 SRPM.os... 等档案就不必建立了！)。好了，那么就来给他安装 APT Server 相关的流程吧！

5. 建置所需的目录与复制所需的档案：
如上面提到的 表一 所示为我们所需要的目录，而在每个 RPMS.os 为系统原本的光盘里面的档案，而 RPMS.updates 则为套件升级版本。我这里主要以中山大学的 FTP 网站作为主要的 update 数据来源，使用的是 ncftp 的下载方法：

```
1. 建置所需目录：
[root@test root]# mkdir -p /var/www/html/apt/redhat9/RPMS.os
[root@test root]# mkdir -p /var/www/html/apt/redhat9/RPMS.updates
[root@test root]# mkdir -p /var/www/html/apt/mdk9.1/RPMS.os
[root@test root]# mkdir -p /var/www/html/apt/mdk9.1/RPMS.updates

2. 利用原版 CD 来复制所需的 .os 的 RPM 档案
[root@test root]# cd /var/www/html/apt/redhat9/RPMS.os
# 先放入 Red Hat 9 的原版光盘片
[root@test RPMS.os]# mount /dev/cdrom
[root@test RPMS.os]# cp /mnt/cdrom/RedHat/RPMS/* .
[root@test RPMS.os]# umount /dev/cdrom
# 重复上面的步骤，将三片 i386 的 RPM 档案都复制进去！
# 至于 Mandrake 的原版光盘复制方法也是相同的步骤！

3. 利用 ftp.nsysu.edu.tw 来下载所需要的 RPM 档案
[root@test RPMS.os]# cd /var/www/html/apt/redhat9/RPMS.updates
[root@test RPMS.updates]# ncftp  \
> ftp://ftp.nsysu.edu.tw/Linux/RedHat/linux/updates/9/en/os/i386/
NcFTP 3.0.2 (October 19, 2000) by Mike Gleason (ncftp@ncftp.com).
Connecting to 140.117.11.7...
=====================================================================
 欢迎光临【国立中山大学】档案服务器 : FTP.NSYSU.edu.tw
   Welcome to National Sun Yat-Sen University FTP Server.
 右列网址提供本站档案搜寻引擎 http://ftp.nsysu.edu.tw/
   File Search Engine on the URL http://ftp.nsysu.edu.tw/
 目前 FTP 部份有 313 人正在在线，最高限制 5000 人.
   There are currently 313 users out of 5000 possible.
=====================================================================
Only anonymous FTP !!!! Please press [ENTER] key.
Logging in...
 =========================================================================
 !!!! Important Function !!!! (重要功能介绍)
 o 'cd key*word' or 'cd prefix*' or 'cd *suffix' provided (case-insentive).
   (例：cd freebsd 可以打成 cd fr 或 cd *sd 或 cd f*e*d 都通用，大小写都行)
 =========================================================================
 请多多利用 <A HREF="http://ftp.nsysu.edu.tw">http://ftp.nsysu.edu.tw</A>
 可以使用方便的档案搜寻引擎喔!!!!!!!!!!!!!!!!!
 =========================================================================
```

```
 ADM.Email: ftpadm@cc.nsysu.edu.tw
Anonymous user (163.28.112.1) logged in
Logged in to ftp.nsysu.edu.tw.
Current remote directory is /pub/Linux/RedHat/linux/updates/9/en/os/i386.
ncftp ...x/updates/9/en/os/i386 > mget *
.....
```

6. 利用上面的步骤就可以将 Red Hat 9 的两个目录下的 RPM 档案放置完毕！至于 Mandrake 9.1 的方法与 Red Hat 9 是完全相同的！请您依照上面的方法来自行安装 Mandrake 到您的系统当中喔！另外，除了中山大学提供的 FTP 网站之外， 您也可以使用淡江大学或者是其它的大专院校提供的 FTP 服务喔！
   淡江大学的 FTP 网站： ftp://ftp.tku.edu.tw/OS/Linux
   中山大学的 FTP 网站： ftp://linux.cdpa.nsysu.edu.tw

7. 建立 RPM 所在档案的相关数据：
   建立好了档案之后，再来则是需要进行这些 RPM 档案的相关系分析了，分析方法很简单，只要一个动作即可：

```
[root@test root]# genbasedir <最上层目录> <相对目录一> <相对目录二> ...
参数说明：
最上层目录：以我们的 APT 主机为例，最上层目录有两个，分别就是：
            /var/www/html/apt/redhat9
            /var/www/html/apt/mdk9.1
相对目录一：那就是 RPMS.os 与 RPMS.updates ，但是记得 RPMS 不要写，
            只要 os 以及 updates 即可！
范例：
[root@test root]# genbasedir /var/www/html/apt/redhat9 os updates
Creating base directory... done
Components: os updates
Processing pkglists... os updates done
Processing srclists... done
Creating component releases... os updates done
Creating global release file... done
Appending MD5Sum... os updates done
All your base are belong to us!!!
[root@test root]# genbasedir /var/www/html/apt/mdk9.1 os updates
# 动作真是给他有点久～耐心等候吧！ ^_^
```

8. 进行完上面两个动作后，APT 就会在您的系统上面主动的建立起 /var/www/html/apt/redhat9/base 以及 /var/www/html/apt/mkd9.1/base 这两个目录啰！这也是最重要的目录咯！基本上， APT Server 到这一步骤就已经完全 OK 了！其它的就是 WWW 服务器的设定啰，因为前面我们已经介绍过 WWW 了，这里不再重复说明说！

9. 建立 Client 所需要的 sources.list
我们的 Red Hat 9 所架设的 APT Server 当然也可以让我们自己来升级了！这个时候请您修改 /etc/apt/sources.list 呢！请注意喔！您所选择的 APT Server 需要设定正确才行喔！

```
[root@test root]# vi /etc/apt/sources.list
# 这是我们主机的 Red Hat 9 范例：
rpm http://192.168.1.2 apt/redhat9 os updates
# 这是 Mandrake 的范例
# rpm http://192.168.1.2 apt/mdk9.1 os updates

[root@test root]# apt-get update
Get:1 http://192.168.1.2 apt/redhat9 release [543B]
Fetched 543B in 0s (1359B/s)
Get:1 http://192.168.1.2 apt/redhat9/os pkglist [420kB]
Get:2 http://192.168.1.2 apt/redhat9/os release [121B]
Get:3 http://192.168.1.2 apt/redhat9/updates pkglist [55.1kB]
Get:4 http://192.168.1.2 apt/redhat9/updates release [126B]
Fetched 475kB in 8s (57.0kB/s)
Reading Package Lists... Done
Building Dependency Tree... Done
# 这个步骤在测试我们上面的 sources.list 是否正确！
# 并且可以同时更新我们 Client 端的 RPM 属性档案记录！
# 要出现上面的讯息才对，如果出现错误讯息，很有可能是 WWW 设定错误！
```

10. 这里特别说明的是， apt-get update 的作用在『取得 APT Server 的各个 RPM 档案的相关性，亦即是 base 目录里面的档案』，取得这些数据后，未来您的 Linux 主机要进行各项安装/升级动作时，就可以直接取用自己的纪录文件了。所以，如果主机上面更新了 base 里面的信息，则您必须要再次的执行 apt-get update 才行，否则主机上面更新的数据您将无法取得。

11. 定期建立 update RPM 档案的映像数据，并更新 RPM 相关数据
其实到上一步骤所有 APT 相关的作业应该就已经完成了，不过，要晓得的是， Internet 上面的 update 套件是随时在更新的，所以我们的 APT server 上面的 RPMS.updates 目录也应该要随时更新才对啊！要更新，您可以使用手动的方式来下载，用 ncftp 似乎不错！不过，毕竟不太适合实时更新，这个时候，我们可以利用映像 (mirror) 的方式来进行更新喔！我们以 Red Hat 9 的 update 来进行说明！关于映射的说明请您自行参考 http://mirrordir.sourceforge.net/ ，我们这里仅需要应用而已喔！

```
1. 先在线安装 mirror 吧！
[root@test root]# rpm -ivh \
> http://mirrordir.sourceforge.net/mirrordir-0.10.49-1.i386.rpm

2. mirrordir 的语法
```

```
[root@test root]# mirrordir <来源网址> <目标目录>
[root@test root]# mirrordir -v \
> ftp://ftp.nsysu.edu.tw/Linux/RedHat/linux/updates/9/en/os/i386/  \
> /var/www/html/apt/redhat9/RPMS.updates
# 用 -v 来察看一下 mirrordir 的检查状态，如果以 cron 来进行时，就不需要了！

3. 定期进行映像并且同时更新 RPM 档案相关性：
[root@test root]# vi /etc/crontab
# 加入这一行：
30 5 * * * root mirrordir
ftp://ftp.nsysu.edu.tw/Linux/RedHat/linux/updates/9/en/os/i386/  /var/www/html/apt/redhat9/RPMS.
&& genbasedir /var/www/html/apt/redhat9  os updates
# 注意喔！上面为连续的一行啊！ ^_^
```

12.

如此一来，我们的 APT 主机不但能够自己更新自己与 FTP 映像站的 update 数据，并且同时更新 APT 的 base 目录下的相关性档案喔！而您的 APT Client 就可以随时来更新他的 RPM 套件啰！ ^_^

Client 端的设定

无论是 APT Server 或者是 APT Client ，要使用 APT 服务器的功能，您都必须要安装 APT 软件才行。所以，首先请将您的 Client 依据前面『安装 APT 服务器软件』章节进行 apt 的安装；安装完毕之后，最重要的就是修订 /etc/apt/sources.list 这个档案啰！您必须要设定正确的 APT 服务器才行，如此一来，您就可以运用 APT 的强大功能啦！更多的使用技巧请参考 Linux 网络套件升级 章节。

YUM 服务器

谈完了 APT 服务器之后，接下来我们就来谈一谈目前被 Red Hat 及 Fedora 列为预设的 RPM 套件安装/升级机制的 yum 这个服务器啦。

YUM 服务器利用的机制

与 APT 类似的， yum 并没有开发新的网络传输机制，同样仅是利用原本主机就提供的 WWW 或者是 FTP 服务，来让 server/client 进行档案的传输。所以在您 yum 服务器上的 RPM 档案同样的需要放置在 WWW 或 FTP 服务可以存取的所在目录才行。这里鸟哥同样以 /var/www/html 这个 WWW 的目录作为说明。

一般来说，我们需要的 RPM 档案就是原本光盘所提供的套件，以及后来厂商提供的升级套件，这

两种 RPM 档案我分别将他放置在底下所示的目录内：

表二、APT 主机相关 RPM 档案放置的目录示意表

```
/var/www/html/yum/fedora/core1
                            |--base
                            |   `--headers
                            `--update
                                `--headers

/var/www/html/yum/mandarke/10.0
                            |--base
                            |   `--headers
                            `--update
                                `--headers
```

如同上表二所示，每一个版本的 Linux 内仅有两个目录，其中 base 是原版光盘的 RPM 档案，至于 update 则是升级的 RPM 档案。比较有趣的地方在于『经过 yum 分析 RPM 档案后的纪录数据是放置在该目录下的 headers 目录内。』举例来说，我们的 mandrake 10.0 升级用的 RPM 档案是放置在 /var/www/html/yum/mandrake/10.0/update 下，则在该目录下的 RPM 被分析后，每一个 RPM 档案的纪录文件则放置在 /var/www/html/yum/mandrake/10.0/update/headers 目录下喔。至于整个 yum 的机制为：

1. 先将所有来自 Linux 原版光盘的 RPM 档案复制到 /full/path/base 档案中；
2. 再将来自原 Linux 版本公司释出的 RPM 修补套件由 Internet 下载到 /full/path/update 当中；
3. 利用 yum 的功能去分析每个目录下的 RPM 档案；
4. 在 Client 端上面，每次进行 RPM 套件的升级/安装功能时， yum 会自动的读取 headers 内的纪录文件，并自动分析 RPM 套件的属性相依问题。

与 APT 相比， yum 少了一个 Client 端同步化的步骤了，所以，可以避免使用者不小心遗忘了资料同步化而导致无法取得最新 RPM 纪录的问题喔。

安装 yum 软件

yum 在 Red Hat 及 Fedora 是标准配备，所以您无须伤脑筋。但如果您是其它的 distributions 而想要利用这个好用的 yum 功能，那么可以到底下的网站搜寻适合您的 yum 版本：

- http://rpmfind.net/linux/rpm2html/search.php?query=yum
- http://free.tnc.edu.tw/modules/news/article.php?storyid=854

找到后，直接安装即可。

---

yum 服务器的套件结构

鸟哥觉得， yum 服务器的套件结构要比 apt 简单一些些，基本上只有底下几个咚咚：

- 设定档：不要怀疑，只有一个设定档。
    - /etc/yum.conf：这个档案是给 yum client 用的设定档，里面主要规定了要取用 RPM 档案的 yum server 的信息，内容有点像这样：

```
[root@test root]# vi /etc/yum.conf
# 在这个档案当中，批注符号是 # ，而每一个大项目则以 [] 作为开始
# 除了 [main] 是用在针对本机相关参数的说明之外，要取用服务器的目录时，
# 则需要额外的规定。我们以上述的 base/update 两个目录作为说明：
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
pkgpolicy=newest
distroverpkg=fedora-release
tolerant=1
exactarch=1
# 上面主要规定了执行 yum 时所会使用到的目录。例如 /var/cache/yum。

[base]
name=Fedora Core $releasever - $basearch - Base
baseurl=http://127.0.0.1/yum/fedora/core1/base
# 这里就重要了，那个中括号[]里面就是『目录名称』需要对应正确喔
# name 仅只是说明该目录下的咚咚而已，并不重要；
# baseurl 则是完整的 URL 了！这里请千万填写正确！

[update]
name=Fedora Core $releasever - $basearch - Released Updates
baseurl=http://127.0.0.1/yum/fedora/core1/update
```

    - 看到了吗？事实上，我们只要设定好 [base] 与 [update] 里面的网址，呵呵！就可以使用 yum server 所提供的更新功能啦。

- 执行档：
    - yum：这个指令是给 yum client 用来作为更新之用的，简单的操作如下：

```
[root@test root]# yum <options> <更新项目> <套件名称>
参数说明：
options：这里仅列出常见的参数而已。
    -y 如果 yum 在工作过程中需要使用者响应，这个参数可以直接回答 yes
更新项目：更新的动作有底下几个：
    install ：安装某个套件，后面需要接套件名称；
    update  ：这就是升级啦！如果 update 后面接套件名称，表示只要 yum 升级该套件
              而已，如果 update 后面不接套件名称，表示 yum 针对目前该主机所有已
              经安装的套件进行升级的动作！这是最常使用的项目了。
    list    ：列出目前在 yum server 上面有的 RPM 套件；
    info    ：某个套件的内容，类似 rpm -qi packages 的内容。
    clean   ：将已将下载到本机的 packages 或 headers 移除。
    remove  ：移除已经安装在我们系统的某个套件！
范例：
[root@test root]# yum install hdparm  # 安装 hdparm 这个套件
[root@test root]# yum update hdparm    # 更新 hdparm 这个套件
[root@test root]# yum -y update      # 更新目前本机上面的所有套件，并自动回复 yes
[root@test root]# yum clean packages # 将下载至本机的 RPM 档案删除(放在
                                          /var/cache/yum 里面。

# 至于每日更新的话，可以写入 /etc/crontab 喔
[root@test root]# vi /etc/crontab
40 5 * * * root yum -y update; yum clean packages
```

-
- yum-arch：这个指令则是给 yum server 使用的！重点在分析 RPM 套件的 header 喔！
  用法真是很简单，同样已 /var/www/html/yum/fedora/core1/base 为例：

```
[root@test root]# yum-arch <options> <目录>
参数说明：
options：这里仅列出常见的参数而已。
    -q : yum 分析 RPM 过程中不显示讯息。
范例：
[root@test root]# yum-arch /var/www/html/yum/fedora/core1/base
# 只要经过这个步骤， yum 就会自动在 /var/www/html/yum/fedora/core1/base 底下新增
# 一个名为 headers 的目录，并将分析的 RPM 纪录数据都放置在里面喔！
```

-

- 相关目录：

  - /var/cache/yum：我们在 yum 这个指令里面谈到当使用 yum 进行升级时，下载下来的
    rpm 档案并不会自动被删除的，那么这些 rpm 档案放在哪里呢？就是放在

/var/cache/yum 这个目录内。这些档案在升级完成后就可以移除了，所以我们可以使用『yum clean packages』来移除这些 rpm 档案喔。

yum 相关的咚咚就只有这样，是否真的很简单啊？ ^_^

---

yum 服务器设定

yum 服务器的设定真是简单！最重要的是 WWW/FTP 的设定必须要正确才行。整个步骤是这样的：

```
1. 先建立所需要的目录：
[root@test root]# mkdir /var/www/html/yum/fedora/core1/base
[root@test root]# mkdir /var/www/html/yum/fedora/core1/update

2. 复制原版光盘的内容：
# 利用 mount 与 cp umount 等功能，将原本光盘的内容整个复制到
# /var/www/html/yum/fedora/core1/base 当中。

3. 利用 mirrordir 下载升级套件。鸟哥这里建议使用中山大学的 FTP 网站：
[root@test root]# mirrordir -v \
> http://linux.cdpa.nsysu.edu.tw/Linux/Fedora/linux/core/updates/1/i386/ \
> /var/www/html/yum/fedora/core1/update

4. 进行 RPM 套件分析：
[root@test root]# yum-arch /var/www/html/yum/fedora/core1/base
[root@test root]# yum-arch /var/www/html/yum/fedora/core1/update

5. 为 client 建立 yum.conf 内容：
[root@test root]# vi /etc/yum.conf
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
pkgpolicy=newest
distroverpkg=fedora-release
tolerant=1
exactarch=1

[base]
name=Fedora Core $releasever - $basearch - Base
baseurl=http://127.0.0.1/yum/fedora/core1/base

[update]
name=Fedora Core $releasever - $basearch - Released Updates
baseurl=http://127.0.0.1/yum/fedora/core1/update
```

```
6. 开始自我升级：
[root@test root]# yum -y update
```

没错！别怀疑！整个 yum 服务器的架设就是这么简单啊！

特殊案例

建立自己的更新套件

如果您曾经自己修改一些 SRPM 的档案，并且重新打包成为 RPM 套件，然后发行给自己的 Linux 机器来安装，此外，您也可能取得一些计划所释出的 RPM 档案，例如 opnewebmail 所释出的 RPM 套件，这些 RPM 档案您又想放置在自己的服务器上面，以方便自己未来升级与查询之用，那么应该怎么放置呢？最简单的方法就是将这些档案放置在 /var/www/html/yum/fedora/core1/update 里面，不过可能会有问题，例如如果您使用 mirrordir 进行数据的映像时，那么您放置的 RPM 档案将会被删除喔！

事实上，我们可以自己设定好额外的升级目录啊！举例来说，鸟哥自己所制作出来的 RPM 档案都是放置在：
/var/www/html/yum/fedora/core1/myself
当然，该目录是自己建立的，然后将自己新增的 RPM 档案通通给他复制进去该目录下，之后，就是制作 header 档案啦：
yum-arch /var/www/html/yum/fedora/core1/myself
最后只要在 /etc/yum.conf 里面新增一个目录：
[myself]
name=My personal RPM files
baseurl=http://127.0.0.1/yum/fedora/core1/myself
当然啦，要利用您的 RPM 档案的 client ，他的 /etc/yum.conf 就需要加入上面的设定，(注意：127.0.0.1 是在自己的机器上面跑的缘故，您必须要填写正确的 yum server 的主机名称或 IP 才行。)如此一来，您的 RPM 档案就可以被利用啦！

如何取得网络上的更新组件

上面提到的服务器几乎都是以 FTP 的方式在 internet 上面取得最新的 RPM 套件档案。那如果您的内部网域本身并不提供 FTP 对外联机时，该如何是好？鸟哥曾经遇过一个特别的状况，在该网域内仅提供 port 80 的对外联机，除此之外，一切都是关闭的！而我们使用的 mirrordir 及 ncftp 都是以 ftp 来对外联机，这该如何是好？

好在天无绝人之路啊！记得鸟哥在前面 网络升级套件 那个章节里面提到的 zzgetrpm.sh 档案吗？您只要下载该档案，然后填写正确的 URL ，就可以透过 port 80 下载网络上最新的更新套件了！而后在内部网域当中，您当然就可以透过各方式来进行 yum/apt 的升级噜！^_^

主机的规划技巧与建议

那么架设一部 APT/YUM 主机需要注意哪些事项呢？因为您的 APT 主机还需要运行 WWW ，所以您可能需要独立出一个 partition 以进行虚拟主机的设定，或者是在 /var/www/html 这个预设的 WWW 主页的硬盘空间要足够！反正 APT 主机需要注意的地方最主要还是在于硬盘的空间了！此外， APT 主机最好可以放置在对外频宽较高的网段内，因为还需要对 Internet 取得 update 的 RPM 档案嘛！

而如果您的 APT 主机仅作为 APT 的 update 之用时，而 WWW 仅是附属的功能，那么您在安置 RPM 的所在目录最好额外的限制使用者的浏览网段，避免被外部的人(来自 Internet)作为联机升级的主要主机，那可能会占据掉您的连外频宽呐！所以，可能的话，使用虚拟主机，并且加以设定浏览的属性，例如使用 iptables 设定防火墙，或以 httpd.conf 里面的 ACCEPT/DENY 功能来抵挡，也是一个不错的想法喔！至于 WWW 主机的相关设定请参考前面的章节。

参考资源：

- http://bazar.conectiva.com.br/~godoy/apt-howto/
- http://freshrpms.net/apt/server/
- http://rpmfind.net/linux/rpm2html/search.php?query=apt
- http://163.19.59.1/~linux/student_samba/apt/apt_server.html
- http://cle.linux.org.tw/~candyz/
- http://linux.duke.edu/projects/yum/

简易 OpenWebMail 设定

最近更新日期：2004/11/23

在 Client 端使用 pop3 之类的 MUA 软件来收信是目前一般使用者最常见的信件收受模式。不过， 这种模式由于是直接将信件下载到客户端的个人计算机上，事实上， 很容易遭受不明病毒的影响，并且，如果信件内容大部分是广告信件的话，由主机传送到个人计算机这过程中的传送行为， 将会平白的让使用者损失一定程度的网络频宽。由于电子邮件对于现代人来说， 应该已经是不可或缺的使用工具，所以虽然使用 pop3 有一定程度的风险，大家还是得继续使用啊~难道没有解决的方法吗？呵呵！是有的，那就是使用网页接口 (Web) 的邮件功能啦，也就是 WebMail 。其实 WebMail 并不是一个邮件服务器，而只是透过主机提供的 Web 接口让使用者登入，并且直接在主机上面进行邮件的收发而已。因此，事实上，您也可以将 webmail 视为一个 MUA 啦！底下我们就来说一说这一个由台湾人发明的 OpenWebMail 吧！

前言

电子邮件对于一般人来说，是越来越重要了，像鸟哥通常就是使用电子邮件来跟大部分的朋友联络， 而且，重要的文件与公事也几乎是使用电子邮件来传递的呢！而在一般的个人计算机上面，我们通常是使用类似 Outlook express/Netscape/Mozilla 之类的 Client 端软件，并透过 POP3/SMTP 协议来收发信件。

不过，遗憾的是，电子邮件其实并不安全！这点在一些 安全通报 上面就可以查阅的到了，不论是 MTA/MUA 都可能有漏洞的！而且，除了漏洞问题之外，更麻烦的是，广告信件的大量发行， 造成我们客户端的频宽浪费，真的是很讨厌！但是，电子邮件又是这样的重要， 不能不收发电子邮件啊！那怎么办？这个时候我们就可以使用 Web 接口的 MUA 来进行电子邮件的收发啊！

那么这种 webmail 的地位是甚么呢？是否只要有 webmail 就不需要 mail server 了呢？当然不是这样！ webmail 其实可以被当成是一种 MUA 来看待，也就是说，您可以将他想成与 outlook express 同等级的应用软件；而透过这个 webmail 我们可以对 mail serve r进行电子邮件的存取。也就是说， webmail 要能够使用，必须要架构在具有 WWW 及 Mail Server 的服务上面才行！这几者之间的关系可以简单的这样看：

```
          Server 端                  Client
    -------------------------
   |      Apache/Webmail<---|<----  Mozilla/IE(浏览器)
   |                |             |
   |<---SMTP-----|             |
```

```
|              |            |
|  /var/spool/mail/accout |
`__________________________,
```

也就是说，当 Client 端以浏览器登入主机后，主机透过 webmail 提供的服务，让登入的使用者可以读取该主机内自己的邮件，并且，也可以透过主机的 smtp 协议来进行邮件的寄发！

架设前须知

如同前面的说明，webmail 要能够正确的运作，其实还需要 www/mail 这两种服务的存在才行！此外，由于 webmail 需要能够正确的存取主机内的邮件档案 (/var/spool/mail/account)，所以，我们必须要提供 webmail 足够的权力来存取才行。因此，您必需要确定：

- WWW Server (Apache) 已经顺利运作；
- Mail Server (Postfix/sendmail) 已经顺利运作；
- webmail 要求的前置软件必须已经安装：例如 openwebmail 需要 perl / suidperl / perl(CGI) / perl-Text-Iconv / perl-Compress-Zlib 等等的套件呢！其中 suid 方面的软件，则是提供 webmail 存取权力的套件。

所以啰，在您架设 webmail 之前，请先确认您的主机提供的 WWW/Mail 服务是正确无误的！此外，前驱套件也必须要没有问题的安装才行！

底下我们以目前相当热门，并且是由台湾成大博士董仲恺主导发起的 Openwebmail 这个套件来进行安装！这东西很不错喔！您可以在底下的连结当中下载与 Red Hat/Fedora 有关的 RPM 档案：

- http://turtle.ee.ncku.edu.tw/openwebmail/download/redhat/rpm/packages/

或者是下载源文件 (tarball)：

- http://turtle.ee.ncku.edu.tw/openwebmail/download/release/

底下鸟哥将以 FC1 及 Mandrake 10.0 作为测试的基准来安装 openwebmai 喔！

OpenWebMail 的安装：

因为 openwebmail 本身就有针对 Red Hat/Fedora 释出 RPM 档案，所以我们当然就直接以编译好的 RPM 档案来安装即可！那如果是非 Red Hat/Fedora 怎么办？没关系，可以使用原始码来安装啊！

FC1 的 RPM 安装

请先确定 Postfix 或者 sendmail 已经安装完毕，此外， Apache 也已经安装好了！如果想要快速的架设好您的 Postfix + SASL 认证的话，可以这样做：(更详细的信息请查阅：
http://linux.vbird.org/linux_server/0390postfix.php)

```
# 0. 先确定所有的套件都已经安装了，主要有:
     postfix-2.0.16-1
     cyrus-sasl-2.1.15-6
     cyrus-sasl-md5-2.1.15-6
     cyrus-sasl-devel-2.1.15-6
     cyrus-sasl-plain-2.1.15-6
     如果还有忘记写上去的，还请继续的安装啊！并且，请移除底下的套件
     sendmail
     fetchmail
     mutt

# 1. 先确定一下 /usr/lib/sasl2/smtpd.conf 存在，且内容为：
     pwcheck_method: saslauthd

# 2. 确定一下 /etc/sysconfig/saslauthd 内容为：
     MECH=shadow

# 3. 确定一下 main.cf 里面有：
smtpd_sasl_auth_enable = yes
smtpd_sasl_local_domain=
smtpd_recipient_restrictions =
  reject_unknown_sender_domain
  reject_unknown_recipient_domain
  permit_mynetworks
  permit_sasl_authenticated
  reject_unauth_destination
smtpd_client_restrictions =
  permit_mynetworks
  hash:/etc/postfix/access
  permit_sasl_authenticated
  reject_rbl_client relays.ordb.org
  reject_rhsbl_client dsn.rfc-ignorant.org
smtpd_sasl_security_options = noanonymous

# 4. 然后启动 saslauth 及 postfix 应该就可以啦！
```

然后，请先下载 FC1 所需要的前驱套件，请连结至：
http://turtle.ee.ncku.edu.tw/openwebmail/download/redhat/rpm/packages/fc1/ 下载所有的 RPM 套件，然后再安装 openwebmail 2.40 版，整个安装流程：

```
1. 安装 openwebmail ：
[root@test root]# rpm -ivh perl* # 假设所有的 RPM 都放在 /root 底下。
[root@test root]# yum update
# 这个动作仅只是要升级刚刚的套件而已；
```

```
[root@test root]# rpm -ivh openwebmail-2.40-1.i386.rpm
# 注意：在预设的情况下， openwebmail 会将档案资料安装至与 apache 有关的路径
# 也就是 /var/www/data 与 /var/www/cgi-bin 底下，所以，如果您已经将 apache
# 以 tarball 方式安装，导致拥有不同的 WWW Root 路径时，最好使用 tarball
# 来安装您的 openwebmail 啊！

2. 初始化设定：
[root@test root]# cd /var/www/cgi-bin/openwebmail
[root@test openwebmail]# ./openwebmail-tool.pl --init
# 因为 openwebmail 支持相当多种认证机制，因此，我们必须在使用 openwebmail
# 之前，先将 openwebmail 的相关设定做好！上面 openwebmail-tool.pl --init
# 就是在达成这样的目标。此外，这个步骤会进行蛮长的一段时间喔！请耐心等候！

[root@test openwebmail]# cd etc
[root@test etc]# vi dbm.conf
# 找到并修改成底下几行：
dbm_ext                 .db
dbmopen_ext             .db
dbmopen_haslock         yes
[root@test etc]# cd ..
[root@test openwebmail]# ./openwebmail-tool.pl --init
Welcome to the Open WebMail!

This program is going to send a short message back to the developer,
so we could have the idea that who is installing and how many sites are
using this software, the content to be sent is:

OS: Linux 2.4.22-1.2199.nptl i686
Perl: 5.008003
WebMail: Open WebMail 2.40 20040816

Send the site report?(Y/n)
sending report...

Thank you.


3. 进一步设定：
[root@test openwebmail]# cd etc
[root#test etc]# vi openwebmail.conf
# 找到这一行：
default_language                en
# 改成这样子：
```

```
default_language                zh_TW.Big5
```

一般来说，设定成这样就足够您运作 openwebmail 了，不过，您的 WWW 主机必须要能提供 perl 的执行环境，因为 openwebmail 就是以 perl 写成的啊！以 apache 为例，您必须要知道如何启动 perl 的 CGI 执行环境：

```
[root@test root]# vi /etc/httpd/conf/httpd.conf
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
AddHandler cgi-script .cgi .pl
```

至少上面的设定要设定妥当才行啊！好了，那么如何使用 openwebmail 呢？直接在您的浏览器上面输入：

http://your.linux.server/cgi-bin/openwebmail/openwebmail.pl

就可以使用 openwebmail 啰！很简单吧！不过，如果万一您的认证一直无法成功的话，可以指定一下认证机制看看：

```
[root@test root]# cd /var/www/cgi-bin/openwebmail/auth
[root@test auth]# vi auth_unix.pl
# 找到底下：
my $passwdfile_plaintext = $conf{'passwdfile_plaintext'} || '/etc/passwd';
my $passwdfile_encrypted = $conf{'passwdfile_encrypted'} || '/etc/master.passwd';
# 改成这样：
my $unix_passwdfile_plaintext="/etc/passwd";
my $unix_passwdfile_encrypted="/etc/shadow";

# 或者是修改底下这个档案成为：
[root@test root]# vi \
/var/www/cgi-bin/openwebmail/etc/defaults/auth_unix.conf
# 其内容只需将
passwdfile_encrypted /etc/master.passwd
# 改为
passwdfile_encrypted /etc/shadow
# 这样就算完成了。
```

MDK10.0 的 Tarball 安装

如果您是使用 MDK 作为您的 Linux 系统时，由于 OpenWebMail 并没有针对这个操作系统来设计 RPM 档案，所以，这里我们以 Tarball 来进行安装。首先，请先确认一下您的 Postfix 与 Apache 已经安装妥当。底下我们就完整的来谈一谈啰！(同样的，更详细的 Postfix + SASL 的 SMTP 身分认证信息请查阅：http://linux.vbird.org/linux_server/0390postfix.php)

```
# 0. 先确定一下底下这些套件都已经安装了(使用 rpm 的方式来安装的)
[root@test root]# rpm -qa | egrep '(sasl|postfix|imap)'|sort
cyrus-sasl-2.1.15-10mdk
imap-2002d-8mdk
libpostfix1-2.1.1-0.1.100mdk
libsasl2-2.1.15-10mdk
libsasl2-devel-2.1.15-10mdk
libsasl2-plug-anonymous-2.1.15-10mdk
libsasl2-plug-login-2.1.15-10mdk
libsasl2-plug-plain-2.1.15-10mdk
postfix-2.1.1-0.1.100mdk
# 若没有安装任何一个套件，请使用 urpmi 来安装喔!

# 1. 建立 saslauthd 认证机制与 smtp 认证档案:
[root@test root]# vi /etc/sysconfig/saslauthd
#SASL_AUTHMECH=pam
SASL_AUTHMECH=shadow
# 事实上，就是将认证机制修订成为 shadow 就是了。

[root@test root]# mkdir /etc/postfix/sasl
[root@test root]# vi /etc/postfix/sasl/smtpd.conf
pwcheck_method: saslauthd
mech_list: plain login
# 这一版的 postfix 比较奇怪，他的认证档案预设就是放置在这里喔!

# 2. 修订 postfix 的设定档 master.cf:
[root@test root]# vi /etc/postfix/master.cf
# 找到这个:
smtp    inet    n       -       y       -       -       smtpd
# 改为这个:
smtp    inet    n       -       n       -       -       smtpd

# 3. 修订 postfix 的设定档 main.cf:
# 基本上，内容就有点类似底下这样:
# Postfix 在 MDK 10.0 底下的预设设定数据:
readme_directory = /usr/share/doc/postfix-2.1.1/README_FILES
sample_directory = /usr/share/doc/postfix-2.1.1/samples
html_directory = /usr/share/doc/postfix-2.1.1/html
sendmail_path = /usr/sbin/sendmail.postfix
setgid_group = postdrop
command_directory = /usr/sbin
manpage_directory = /usr/share/man
daemon_directory = /usr/lib/postfix
```

```
newaliases_path = /usr/bin/newaliases.postfix
mailq_path = /usr/bin/mailq.postfix
queue_directory = /var/spool/postfix
mail_owner = postfix

# 使用者的设定数据！
mynetworks_style = host
delay_warning_time = 4h
smtpd_banner = $myhostname ESMTP $mail_name ($mail_version) (Mandrake Linux)
unknown_local_recipient_reject_code = 450
smtp-filter_destination_concurrency_limit = 2
lmtp-filter_destination_concurrency_limit = 2
smtpd_sasl_path = /etc/postfix/sasl:/usr/lib/sasl2  # <== 就是这一行
smtpd_sasl_application_name = smtpd
relayhost = [seed.net.tw]
#上面这行很重要，如果想要以动态 IP 架站，就需要 ISP 的 relay host 才行！
#详细资料请参考：(因为我的主机在 seednet 底下，所以....)
# http://phorum.study-area.org/viewtopic.php?t=18621
# http://phorum.study-area.org/viewtopic.php?t=22806
myhostname = hostname.domain.name  # <==这里请输入您的主机名称
mydomain = domain.name              # <==这里则是领域名称
myorigin = $myhostname
inet_interfaces = all
mydestination = $myhostname
mynetworks = 127.0.0.0/8
alias_maps = hash:/etc/postfix/aliases
alias_database = hash:/etc/postfix/aliases
mail_spool_directory = /var/spool/mail
debug_peer_level = 2
debugger_command =
         PATH=/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin
         xxgdb $daemon_directory/$process_name $process_id & sleep 5
smtpd_sasl_auth_enable = yes
smtpd_sasl_local_domain =
smtpd_recipient_restrictions =
     reject_unknown_sender_domain
     reject_unknown_recipient_domain
     permit_mynetworks
     permit_sasl_authenticated
     reject_unauth_destination
smtpd_client_restrictions = hash:/etc/postfix/access, permit_sasl_authenticated,
     reject_rbl_client   relays.ordb.org,
     reject_rhsbl_client dsn.rfc-ignorant.org
```

```
smtpd_sasl_security_options = noanonymous
notify_classes = resource, software
message_size_limit =   31457280
mailbox_size_limit = 1000000000
header_checks = regexp:/etc/postfix/header_checks
body_checks = regexp:/etc/postfix/body_checks

# 4. 其它相关的档案:
/etc/postfix/access
/etc/postfix/header_checks
/etc/postfix/body_checks
/etc/postfix/aliases
# 相关设定请参考上面提供的连结啊!

# 5. 开始启动:
[root@test root]# /etc/rc.d/init.d/saslauthd start
[root@test root]# /etc/rc.d/init.d/postfix   start
```

这样就完成了 Postfix 的安装与设定，再接下来则是 Apache 的安装与设定了。

```
# 0. 利用 urpmi 来安装所需要的所有的套件!
[root@test root]# urpmi apache2 php MySQL
Preparing...                ###########################################
   1:libmysql12             ###########################################
   2:libapr0                ###########################################
   3:apache-conf            ###########################################
   4:apache2-common         ###########################################
   5:perl-Data-ShowTable    ###########################################
   6:perl-Mysql             ###########################################
   7:apache2-modules        ###########################################
   8:apache2                ###########################################
   9:libphp_common432       ###########################################
  10:perl-CGI               ###########################################
  11:MySQL-client           ###########################################
  12:MySQL-common           ###########################################
  13:php-ini                ###########################################
  14:apache2-mod_php        ###########################################
  15:MySQL                  ###########################################
# 因为我再最早安装时，并没有选择 WWW ，所以要安装的数据就很多!
# 另外，请您做好 urpmi 的设定才行!

# 1. 开始启动 httpd 啰!
[root@test root]# /etc/rc.d/init.d/httpd start

# 2. 让 MySQL 在开机时不会自动启动， httpd 则会自动启动!
```

```
[root@test root]# chkconfig httpd on
[root@test root]# chkconfig mysql off
[root@test root]# chkconfig --list | grep 3:on
httpd           0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

千万注意，我们目前并没有要使用 MySQL 的计划，所以，当然就要将 MySQL 给他关闭啦！而 httpd 则是需要在开机的时候就与以启动才行， 所以上面的第二个步骤才需要如此的设定喔！现在，您就可以在您的网址列输入您的主机名称，看看能不能连结 http 提供的 WWW 服务呢？！

现在，请注意，预设的状态下， MDK 10.0 的 WWW 主网页是放置在 /var/www/html 底下的，但是 MDK 却没有预设的 data 目录，所以我们就必须要给予设定啦！您可以这样做：

```
# 0. 将 openwebmail 解压缩，并且移动到适当的目录下：
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /disk1/software/openwebmail-2.40.tar.gz
[root@test src]# cd cgi-bin/
[root@test cgi-bin]# mv openwebmail/ /var/www/cgi-bin/
[root@test cgi-bin]# cd ..
[root@test src]# rmdir cgi-bin
[root@test src]# mv data/openwebmail /var/www/html/openwebmail_data
[root@test src]# rmdir data
# 同时请注意，您的主机上面必须要有 perl 模块的存在了才行！
# 我的 perl 模块有底下这么多：
perl-MailTools-1.59-2mdk
perl-Crypt-SSLeay-0.51-3mdk
perl-Term-Readline-Gnu-1.14-5mdk
perl-Digest-SHA1-2.04-1mdk
perl-Mysql-1.22_19-9mdk
perl-MDK-Common-1.1.11-2mdk
perl-HTML-Parser-3.35-2mdk
perl-File-Slurp-2002.1031-3mdk
perl-Tk-800.024-4mdk
perl-SGMLSpm-1.03ii-6mdk
perl-SDL-1.20.0-8mdk
perl-ldap-0.31-2mdk
perl-Glib-1.021-3mdk
perl-Gtk2-1.023-3mdk
perl-IO-Tty-1.02-9mdk
perl-Gtk2-TrayIcon-0.03-3mdk
perl-Authen-SASL-2.04-2mdk
perl-Locale-gettext-1.01-11mdk
perl-TimeDate-1.16-3mdk
perl-HTML-Tagset-3.03-7mdk
perl-URI-1.25-1mdk
perl-Gnome2-Vte-0.02-1mdk
```

```
perl-devel-5.8.3-5mdk
perl-XML-SAX-0.12-3mdk
perl-Convert-ASN1-0.16-4mdk
perl-Data-ShowTable-3.3-9mdk
perl-5.8.3-5mdk
perl-DBI-1.40-2mdk
perl-Libconf-0.33-2.1.100mdk
perl-XML-Parser-2.34-1mdk
perl-CGI-3.00-2mdk
perl-URPM-0.94-11mdk
perl-Expect-1.15-6mdk
perl-Digest-HMAC-1.01-11mdk
perl-base-5.8.3-5mdk
perl-libwww-perl-5.76-3mdk
perl-XML-NamespaceSupport-1.08-3mdk
# 另外，也需要加入两个必要的模块，您可以使用 FC1 的相关档案，下载点：
# http://turtle.ee.ncku.edu.tw/openwebmail/download/redhat/rpm/packages/fc1/
# rpm -ivh perl-Text-Iconv-1.2-fc1.i386.rpm perl-Compress-Zlib-1.16-12.i386.rpm
# 两个就可以啦！

# 1. 初始化设定：
[root@test root]# cd /var/www/cgi-bin/openwebmail
[root@test openwebmail]# ./openwebmail-tool.pl --init
# 因为 openwebmail 支持相当多种认证机制，因此，我们必须在使用 openwebmail
# 之前，先将 openwebmail 的相关设定做好！上面 openwebmail-tool.pl --init
# 就是在达成这样的目标。此外，这个步骤会进行蛮长的一段时间喔！请耐心等候！
[root@test openwebmail]# cd etc
[root@test etc]# vi dbm.conf
# 找到并修改成底下几行：
dbm_ext                 .pag
dbmopen_ext             none
dbmopen_haslock         yes
[root@vbird etc]# vi openwebmail.conf
# 找到并修改成底下几行：
domainnames             auto
auth_module             auth_unix.pl
mailspooldir            /var/spool/mail
ow_cgidir               /var/www/cgi-bin/openwebmail
ow_cgiurl               /cgi-bin/openwebmail
ow_htmldir              /var/www/html/openwebmail_data
ow_htmlurl              /openwebmail_data
logfile                 /var/log/openwebmail.log
[root@test etc]# cd ..
```

```
[root@test openwebmail]# ./openwebmail-tool.pl --init

creating db /var/www/cgi-bin/openwebmail/etc/maps/b2g ...done.
creating db /var/www/cgi-bin/openwebmail/etc/maps/g2b ...done.
creating db /var/www/cgi-bin/openwebmail/etc/maps/lunar ...done.
Welcome to the Open WebMail!

This program is going to send a short message back to the developer,
so we could have the idea that who is installing and how many sites are
using this software, the content to be sent is:

OS: Linux 2.6.3-14mdk i686
Perl: 5.008003
WebMail: Open WebMail 2.40 20040816

Send the site report?(Y/n) y
sending report...

Thank you.

# 2. 修订认证机制：
[root@test root]# vi /var/www/cgi-bin/openwebmail/auth/auth_unix.pl
# 找到：
my $passwdfile_encrypted = $conf{'passwdfile_encrypted'} || '/etc/master.passwd';
# 修改成：
my $passwdfile_encrypted = '/etc/shadow';

# 或者是修改底下这个档案成为：
[root@test root]# vi \
/var/www/cgi-bin/openwebmail/etc/defaults/auth_unix.conf
# 其内容只需将
passwdfile_encrypted /etc/master.passwd
# 改为
passwdfile_encrypted /etc/shadow
# 这样就算完成了。
```

然后连到您的主机：http://hostname/cgi-bin/openwebmail/openwebmail.pl ，就能够看到您的 OpenWebmail 啰！同时，如果还是无法看到相关的咚咚，请查阅您的登录档！

一些其它的设定：

除了既有的设定之外，如果您想要其它的设定时，这里提供几个简单的设定：

```
# 1. 网络硬盘：
如果不想提供网络硬盘的话，可以在底下的档案：
```

```
/var/www/cgi-bin/openwebmail/etc/openwebmail.conf
加入这行：
enable_webdisk no

# 2. 提供 ssh 联机：
如果不想提供网络联机的话，可以在底下的档案：
/var/www/cgi-bin/openwebmail/etc/openwebmail.conf
加入这行：
enable_sshterm no

# 3. 让登入更简单：
如果不想在网址列输入长长的一段文字，可以修改成为：
vi httpd.conf (请依照您的系统来修订！)
ScriptAlias  /webmail     /var/www/cgi-bin/openwebmail/openwebmail.pl
or
ScriptAlias  /webmail.pl  /var/www/cgi-bin/openwebmail/openwebmail.pl
然后重新启动 apache ，如此一来，您可以在网址列输入：
http://hostname/webmail or http://hostname/webmail.pl
就可以进入 openwebmail 啰！
```

问题与解决

- 还是无法执行 openwebmail：

  一般来说，安装好 openwebmail 之后，应该是立即可以使用了，不过，如果您一直无法进入 openwebmail 的欢迎画面，那么请特别留意：

  1. 在 openwebmail.conf 这个设定档当中，是否填写了适当的认证机制(auth_unit.pl)；
  2. 在 auth_unix.pl 这个设定档当中，是否填写了正确的密码认证档案？
  3. 您的 WWW 是否提供 perl 的 CGI 执行环境？
  4. 仔细查阅 /var/log/httpd/error_log 及 /var/log/openwebmail.log 来解决问题！

- 来自讨论区的网友留言

```
鸟哥好：
最近刚好在装 openwebmail，拜读主站上的文件安装下去，
发现有一些部份在安装时可能需要额外注意的，在此提供给您。
我的 distribution 是用 MDK 9.2，因此我是照 MDK 10.0 的
Tarball 安装作下去的，不过我的 postfix + LAMP 是用 rpm
装的，因此我直接跳到 openwebmail 安装。
```

```
文中提到：
现在，请注意，预设的状态下， MDK 10.0 的 WWW
主网页是放置在 /var/www/html 底下的，但是 MDK
却没有预设的 data 目录，所以我们就必须要给予设定啦！
您可以这样做：

但我在 ./openwebmail-tool.pl --init 时仍会碰到要找
/usr/local/www 底下的 cgi-bin 的情形，所以应该还是要
 ln -s /usr/local/www /var/www

另外，文件中下达 ./openwebmail-tool.pl --init 两次，
但事实上第一次执行时系统会告诉我 dbm.conf 的信息不对，
所以我认为应该先改 dbm.conf 再执行即可？

以上是我粗浅的看法，其实不管怎么样，可以正确的初始化才是重点。

最后是安装后不能改密码的问题，研究后发现在
cgi-bin/openwebmail/etc/defaults/auth_unix.conf
除了更改 passwdfile_encrypted 外，也要将 passwdmkdb 设定成 none 才行。

这几点是我安装时几个相异之处，提出来供鸟哥参考。也谢谢鸟哥文件的分享。
```

参考资源

- OpenWebMail 的下载： http://turtle.ee.ncku.edu.tw/openwebmail/download/
- OpenWebMail 台湾主站： http://turtle.ee.ncku.edu.tw/openwebmail/
- Ultraman 的 OpenWebMail 实作： http://linux.vbird.org/somepaper/20030317-openwebmail.php

---

简易 Cluster 架设

最近更新日期：2003/05/20

近年来因为数值模式仿真的盛行，所以『平行运算』的架构也就越来越重要了！什么是数值模拟呢？主要就是藉由一些物理理论去开发出来的一些『计算公式』，而这些计算公式藉由程序语言(例如 C、Fortran 等等)实际的将他编译成为可执行的程序，最常见的例如中央气象局不是每天都会预报天气吗？这个预报的动作就是利用数值计算去演算出来的。另外，还有空气质量模式仿真，也是经过运算出来的，除此之外，例如天文、物理、水文等等很多方面的工作，都是利用这种数值模拟的运算的喔！不过，这些程序是很大型的！也就是说，他们在运算的时间是很长的，如果使用单颗 CPU 的话，不论这颗 CPU 的频率与效能有多高，还是得要运算个好几个钟头的～如此一来，对于像气象预报这个急需时效性的工作可能就会有所延误啊！不过，如果我将这个运算的工作同时丢给多颗 CPU 呢？也就是让多颗 CPU 同时进行这个程序的运算工作，如此一来，将可以大大的减低时间的损耗了～这就是平行运算的简单说明。在 Linux 平台上面，要达成简单的平行运算，可以透过 MPI 的函式库，例如 MPICH 就是一个很有名的 MPI 软件喔！马上来给他看看平行运算类型的 Cluster 建置吧！

原理：

- 什么是 Cluster 与 Cluster 的优点

什么是 Cluster 呢？目前常见的 Cluster (丛集)架构有两种，一种是 Web / Internet cluster system，这种架构主要是将数据放置在不同的主机上面，亦即由多部主机同时负责一项服务；而另外一种则是所谓的平行运算了！平行运算其实就是将同一个运算的工作，交给整个 Cluster 里面的所有 CPU 来进行同步运算的一个功能。由于使用到多个 CPU 的运算能力，所以可以加快运算的速度。目前比较常见于平行运算功能的，通常需要在超级计算机上面才看的到，这些超级计算机主要是用在天文、军事、物理等需要很精密的、大量的运算的工作中，而考虑到稳定性，则

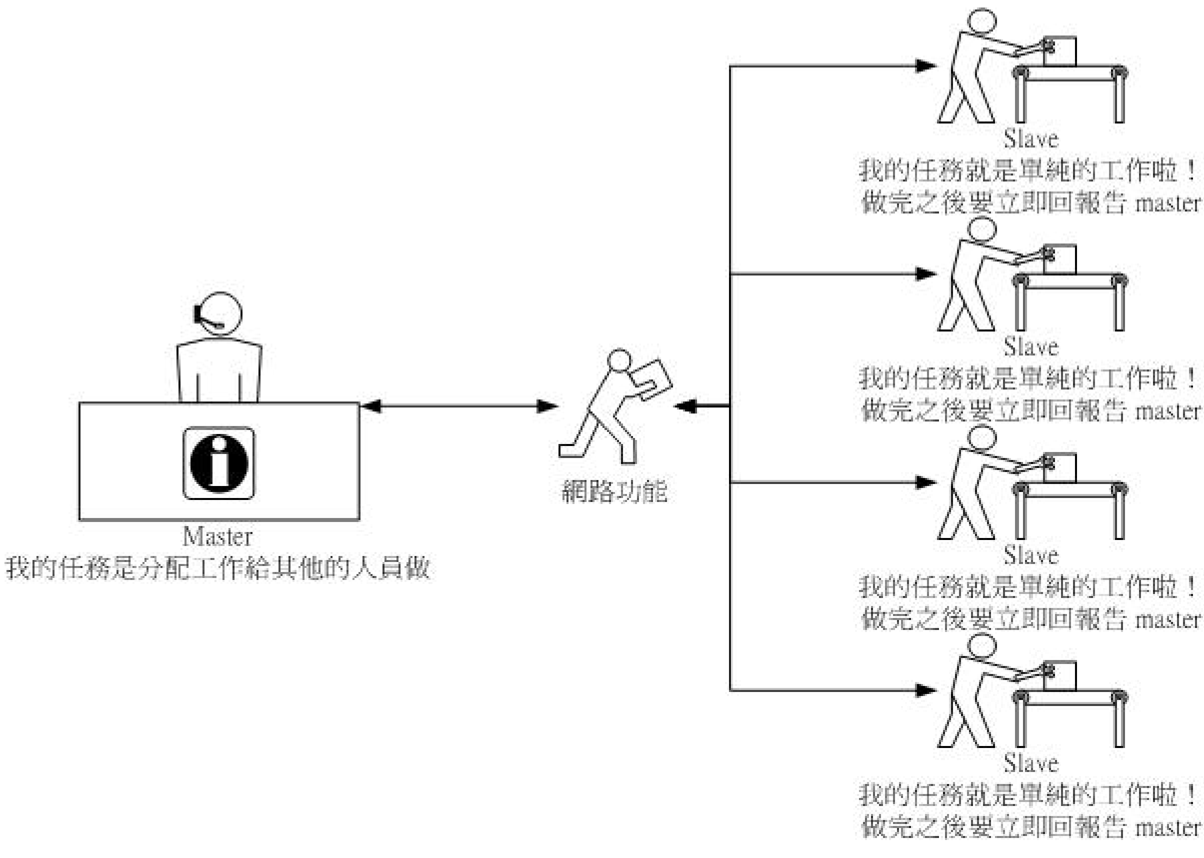
通常是用在 Unix 系统上面的硬件架构上。不过，目前由于 PC 上面的 CPU 的运算功能越来越强大了～因此，当然很多程序开发者就动脑筋到 PC 上面来制作并行计算机的系统啰！我们这篇短文主要在介绍的就是『平行运算』这一类的 Cluster 了！

由于 Cluster 主要是用在平行运算上面的，而所谓的平行运算是使用到多颗 CPU 的运算功能，因此可以让您的大型运算的程序很快的执行完毕！因此，如果你的工作环境当中，常常会使用到很耗 CPU 运算功能的程序时，就可以尝试使用 Cluster 来进行工作啰！应该可以节省您不少的时间呐！此外，我们这篇短文主要是在 X86 架构下的 PC 来架设 Cluster 的喔！

不过，也需要特别留意的是，由于我们的 Cluster 是将一个工作平均分给所有的 node (注：一颗 CPU 在一个 Cluster 架构下，就称为一个 node 啰！)，所以，万一您做成的 Cluster 系统当中，所有的 node 并非完全相同的运算等级，那么先做完工作的 node 将会暂停工作，会等到所有的 node 都进行完毕后，才会在进行下一动～所以啦！强烈的建议在同一个 cluster 的架构中，尽量所有的 CPU 都使用相同的 CPU 型号，应该会比较好一点喔！

- Cluster 的主从架构

最简单的 Cluster 其实就是以一种主从架构来进行数据的运算工作的，图示如下：



- 上面的 Master 与 Slave 指的都是 CPU 喔！

- Master 那部机器上面必须要有可以将工作分配给各个 node 去工作的函式库，也就是 MPI ，他最重要的功能就是将工作给他分配下去的啦！而最重要的软件就是：(1)MPICH; (2)编译器(compiler, 例如 Fortran);

- 什么是网络功能呢？如果 master 与 slave 是在同一部机器当中，例如双 CPU 的主机板，那么这里就不需要网络功能啦！不过，如果我是使用四台双 CPU 的 PC 呢？呵呵！那么这四部主机就需要以高速网络架构进行联机啦！此外，还需要在这四部主机之间建立可以互通讯息的通讯协议才行，这方面的功能就含有：(1)R Shell, 亦即称为 RSH; (2)NIS，使 Master 与 Slave 具有相同的账号群组关系；(3)NFS，使读取写入的数据可以在同一个 partition 上面；

- Slave 就是单纯的将来自 Master 的任务给他做完就是了！

整个主从架构大致上就是这样啦！因此，可以知道的是，我们需要的就是上面那些咚咚啰！

- 达成 Cluster 所需要额外功能 (RSH) 与软件 (MPICH)

由上面的 Cluster 主从架构当中，我们知道 Master 与 Slave 之间的网络沟通很重要的一个咚咚，那就是 R Shell 啰！此外，还有将一个工作传送给不同的 node 来进行计算的任务，就需要 MPICH 这个函式库来进行！简单的谈一谈这两个玩意儿吧！

- RSH：
  在我们的 Linux 主机上面工作，通常使用 BASH 这个 shell 来传达给 kernel 工作的讯息，以使主机正确的工作；而如果在不同的主机之间，那就可以使用 R Shell 来进行指令的下达喔，如此一来，我们就可以直接在 A 机器，向 B 机器下达工作的指令，而不需要登入 B 机器呢～那就是 RSH 的主要功能啦！最常见的 RSH 指令就是 rcp 与 rsh 了！有兴趣的朋友应该知道以 man 来查寻一下该指令的用法啰！
  需要附带一提的是，这个 RSH 是『相当危险』的一个服务喔！由于我们可以直接登入 RSH 主机，并且在上面进行指令的下达，为了避免还要输入密码的问题，因此通常 RSH 已经将信任主机设定好了，不过，由于 RSH 会启动一些 port 来监听 Clients 的需求，而偏偏这些 port 与 daemon 都还挺危险的，因此，『Cluster 最好是设定在内部网域当中，并使用私有 IP ，比较能够避免危险』喔！此外，那个 Master 也必须要设定好一定程度的严密防火墙喔！

- MPICH：
  MPI 是 Messages Passing Interface 的缩写，他本身是一个规格很严密的通讯标准，主要的功能是在处理平行运算之间各个 node 的数据交换，请注意， MPI 并不是一套软件喔！而至于 MPICH 就是符合 MPI 这个标准通讯协议的一套软件了！因此，我们可以经由 MPICH 这个软件提供的 MPI 函式库来达成平行运算的功能喔！也就是说，我们所

写的程序，只要能够使用 MPICH 提供的函式库，那么该程序就可以进行平行运算时候所需要的功能了，这就可以避免程序开发者还要去处理通讯节点上面的问题，而可以将程序开发的重心着重在程序本身的问题上面！
MPICH 是由 Mathematics and Computer Science Division的 Argonne 实验室所发展，详细的数据可以参考：http://www-unix.mcs.anl.gov/mpi/mpich/

除了这两个软件之外，还需要 NIS 与 NFS 喔！所以啦！要建置一个 Cluster 的话，身为系统管理员的您，必须要学会的技能真是相当的多的，至少需要：

- 熟悉 Linux 的操作技巧；
- 熟悉 Linux 基础网络参数设定；
- 熟悉 Linux 相关的 Server 架设(这方面请参考鸟哥的私房菜架站篇)；
- 了解 RSH 的相关功能与设定技巧；
- 了解 MPICH 的设定与相关功能；
- 熟悉至少一种程序语言。

还真的是不好学啊！鸟哥也是新手玩弄 Cluster 说～大家一起研究研究吧！ ^_^

架设流程

要架设 Cluster 当然就是需要多部的 PC 来联机啦！不然怎么称为 Cluster 呢？您说是吧！所以，无论如何，在架设 Cluster 之前，请务必要确认您的『所有硬件以及网络功能都是完整无缺的！』否则就无法继续下去啦！另外，建议 Cluster 的所有主机规格尽量相同，可以避免等待的困扰呢！底下就来谈一谈整个架设流程吧！

- 整体架构

整体架构的架设当中，需要的所有软件为：

- Master 主机安装需要：
    - 防火墙的设定(含 NAT 架设)；
    - RSH
    - NIS Server
    - NFS Server
    - Compiler Install
    - MPICH Install
    - 其它特殊功能

- Slave 主机安装需要：
  - 防火墙的设定
  - RSH
  - NIS Client
  - NFS Client

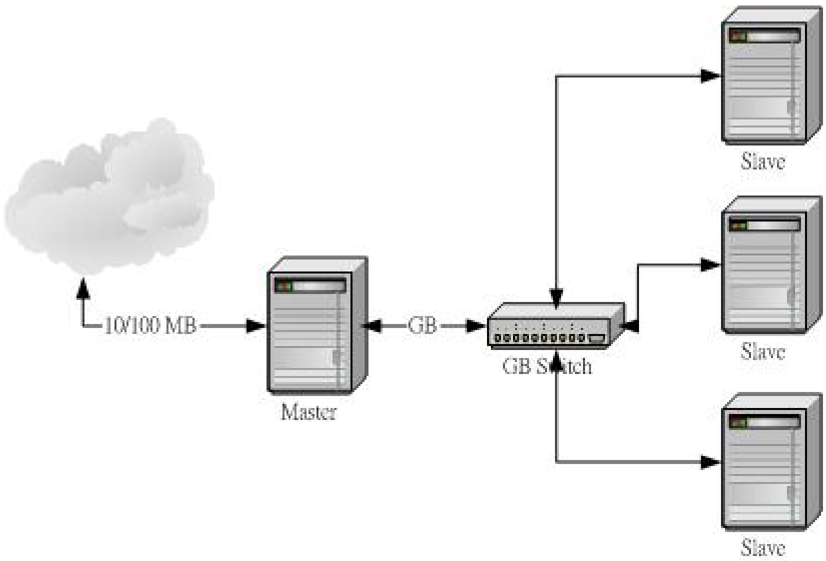
基本上，几乎所有的工作都是在 Master 上面做啦！ Slave 最大的任务就是进行来自 Master 所要求的计算工作，因此，Slave 能够越简单越好～至于 Master 上面，由于我们都是在 Master 主机上面下达工作指令，而总不能老是在屏幕前面下达指令吧！因此上，Master 通常会有两个网络接口，分别是对外的 Public IP 与对内的 Priavte IP。而既然 Master 有提供 Public IP 的设定，自然就比较担心所谓的黑客入侵问题，所以啦，您的 Master 主机，要吗就不要开放 Public IP ，要吗就务必要设定很严密的防火墙，并且不必要的服务就尽量关闭他～毕竟我们的 Cluster 是要用来做为计算运作的，所以不必要的网络协议服务，当然就是关闭他啦！底下鸟哥将以自己的一个实际案例进行说明的啦！参考看看吧！

- 鸟哥的一个实例规范

在我这个案例当中，Cluster 主要的功能为:进行 MM5 这个气象模式的运算以及 Models-3/CMAQ 这个空气质量模式的运算，而由于这两个咚咚都是使用 PGI Fortran 90 做为 Compiler ，因此，我就必须要进行 PGI 的安装啦！而我的硬件架构主要是这样的：

- Master ： 为双 CPU 主机，使用 AMD MP 的 CPU ，并且有一颗 120 GB 的硬盘，此外，由于我的数值模式需要 PGI Fortran ，所以就必须要安装 Server 版的 PGI Fortran 喔！
- Slave ：共有三部 Slave ，每一部均为双 CPU 的 AMD MP 的 CPU ，并且有一颗 120 GB 的硬盘；
- 连接 Master 与 Slave 的为 10/100/1000 的 Switch ，当然，四部主机(1 x master + 3 x slave)都是安装 Intel 的 1GB 网络卡喔！

硬件连接有点像这样：



那么底下就来谈一谈怎么安装他吧！

- 系统安装(Red Hat 9)

  我的这个系统使用的是最新的 Red Hat 出版的 Red Hat 9 ，会用这个玩意儿最大的原因是因为 Red Hat 是目前支持的 Linux 软件最多的一个 Linux Distribution 了，安装他之后，就比较不会欠东欠西的，此外，很多的软件都是以 Red Hat 做为测试的平台，因此我就选择他来做为我的系统平台啊！另外，需要留意的是，由于 Slave 并不需要使用到图形接口的功能，他单纯是用在计算上面，因此我没有在 slave 上面安装图形接口的打算～至于 Master 则安装了 KDE 这个咚咚喔！好了，Linux 的安装相信大家应该都要很熟悉了，所以我就不再谈安装的详细步骤，仅提几个特别需要注意的地方啰：

  - Partition 方面：
    因为我的硬盘实在是在蛮大的，并且在 Slave 上面也是 120 GB 的硬盘，如果不将 Slave 的硬盘使用的话，实在觉得很浪费，因此，一开始我就规划将四部主机的硬盘全部都以 NFS 分享到内部网域当中，而为了避免跟系统的档案放在一起，因此，我就将硬盘分割出除了必要的 partition 之外，其它的就挂载在 /disk1 这个目录当中，四部主机的 parition 都相同，分别是：

    - /　　: 10 GB
    - /var :　5 GB
    - /tmp :　3 GB
    - Swap :　3 GB (因为我每部 Linux 主机上面都有 1.5 GB 的内存)
    - /disk1: 96 GB

- 安装时选择的套件：
  所有的主机都需要底下的套件安装(注：因为原本的笔记记录的很乱，所以如果找不到相同的字眼，那就是我写错啦！)：
  - Administrattion Tools
  - Development Tools
  - Editors
  - Engineering and Scientific
  - FTP Server
  - Kernel Development
  - Network Servers
  - Server configuration Tools
  - Sound and vedio
  - System Tools
  - Text-based Internet
  - Windows File servers

  不过， Master 需要额外再增加 X Window 方面的支持，例如 KDE 与 X-Window System 这两个主要的套件要勾选喔！

-

系统安装大致上就是这些吧，如果有疏漏的，请未来在安装完毕之后，再以原本 Red Hat 9 的光盘来重新安装他吧！反正 Red Hat 系统都是以 rpm 来安装的，挺容易安装的喔！整个安装完毕后，还花不到几分钟呢！

-

- 防火墙 (含 NAT 主机) 与网络设定

由于我们的 Cluster 主要是用在数值运算，因此，当然不需要对外开放网络服务啦！所以，最好就是以私有 IP 来进行网络的设定是比较好的！此外，最好还是要设定好防火墙的啦！我的网域参数预设是这样的：

- Network/netmask：192.168.10.0/255.255.255.0
- Master：(对外)140.116.xxx.yyy；(对内)192.168.10.30, Gateway 为对外的 Gateway 喔！并且有设定 NAT 啊！
- Slave：192.168.10.10, 192.168.10.20, 192.168.10.40 三部，Gateway 为 192.168.10.30

网络参数的各个档案是这样的：

- 各个主机的主机名称请修改：/etc/sysconfig/network
- 各个主机的网络卡设定项目请修改：/etc/sysconfig/network-scripts/ifcfg-eth0
- 各个主机的 DNS 查寻系统请修改：/etc/resolv.conf

- 各个主机的内部主机名称查寻系统请修改：/etc/hosts，我的 /etc/hosts 如下：

  ```
  127.0.0.1       localhost       localhost.localdomain
  192.168.10.10   node1.cluster
  192.168.10.20   node2.cluster
  192.168.10.30   server.cluster
  192.168.10.40   node4.cluster
  ```

而我的每部主机先将所有的网络服务都给他关掉去，仅剩下 SSH 这个网络协议存在而已～所以，我利用 Red Hat 提供的 ntsysv 这个指令来选择开机时启动的项目有：

- atd
- crond
- iptables
- keytable
- network
- random
- sshd
- syslogd
- xinetd

至于防火墙系统的规划上面，由于 Master 主机 (192.168.10.30) 具有 NAT 的功能，所以必须要修改一下他的防火墙机制，因此，就有两个不同的防火墙机制 scripts 啰！另外，由于担心外来的入侵攻击，因此，在这个 Cluster 的系统当中，我的 iptables 防火墙机制是使用『MAC(网络卡卡号)』来做为设定的基准，而不是使用 IP 啊！为什么呢？因为反正我仅允许我自己同网域内的几部计算机连进来而已，当然没有必要针对 IP 啊！所以啰，就必须要收集四部主机的网络卡来允许他进入啰。此外，由于我可能并不在 Cluster 前面操作，因此会启用一两部主机的网络卡卡号，好让他能够进入 Cluster 呦！底下就将我的防火墙机制给他列出来一下：

- Master：

```
#!/bin/bash
# This program is for iptables' rules
# VBird 2003/05/02
#
# 0. PATH and modules
  PATH=/sbin:/bin:/usr/sbin:/usr/bin
  export PATH
  modprobe ip_tables
  modprobe iptable_nat
  modprobe ip_nat_ftp
```

```
  modprobe ip_nat_irc
  modprobe ip_conntrack
  modprobe ip_conntrack_ftp
  modprobe ip_conntrack_irc
#
# 1. clear the rules and make the policys
  iptables -F
  iptables -X
  iptables -Z
  iptables -F -t nat
  iptables -X -t nat
  iptables -Z -t nat
  iptables -P INPUT   DROP
  iptables -P OUTPUT  ACCEPT
  iptables -P FORWARD ACCEPT
  iptables -t nat -P PREROUTING  ACCEPT
  iptables -t nat -P POSTROUTING ACCEPT
  iptables -t nat -P OUTPUT      ACCEPT

#
# 2. NAT services
  echo "1" > /proc/sys/net/ipv4/ip_forward
  iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth1 -j MASQUERADE
#
# 3. Trust network and conditions
  iptables -A INPUT -m state --state ESTABLISHED,RELATED  -j ACCEPT
  iptables -A INPUT -i lo -j ACCEPT
  iptables -A INPUT -m mac --mac-source XX:YY:ZZ:WW:QQ:PP -j ACCEPT
# 上面这一行就是网络卡的卡号啦!
```

-
- Slave：
  Slave 的防火墙机制跟 Master 几乎一模一样，只是因为在内部啊，所以不需要启动 NAT 的服务即可！上面的给改一改先～

好啦！网络的设定与防火墙就到这里为止，要记得喔，你的网络必须要已经能够正确的启动了！如果还是无法启动网络，或者是防火墙机制还是有问题，那么对外的那个网络卡的网络线还是先给他拔掉吧！比较安全一些些的啦！等到都设定妥当，尤其是防火墙，然后才来启动他吧！

- NFS 架设规划

  由于我这里预计要设定 NIS ，并且每部主机的 /disk1 都要分享出去，因此，每部主机都必须要开放 NFS 的服务喔！并且，每一部主机的设定都可以相同呐！这样比较容易来设定啰～此外，比较不一样的地方在于 Master 这一部，由于我的 Cluster 所有的账号都在 NIS 的管制之中，因此，我将 Master 的 /home 也分享出来，并且每部 Slave 主机都挂载 Master 的 /home 才成！

  这个 NFS 在 Cluster 当中是相当重要的，为什么呢？因为我们不是在四部主机上面工作吗，而这四部主机会去读取的『数据』其实都是『在本机上面可以看的到的』数据才行，这还包括底下我们会持续介绍的 mpich 这个软件的函式库呢！也就是说：『在 Cluster 里面，所有的机器会使用到的数据必须都在相同的目录当中！』所以，这就是为什么我们要对 /home 进行分享，以及进行 NIS 的设定了！此外，因为我的 Server 这部 Master 机器分享出去的目录中，已经含有 /disk1 这个 partition，此外，还通通将他挂载在 /cluster/server 底下，因此，可以建议：『未来在安装所有的 Cluster 需要的套件数据时，例如 Compiler 以及 MPICH 等等，都可以安装到 /cluster/server 这个目录底下，以使所有的主机都能够使用同一个 partition 来源的数据喔！』

  设定程序：

  - Master：

```
1. 启动 portmap 并且设定开机启动：
[root @server root]# /etc/rc.d/init.d/portmap start
[root @server root]# chkconfig --level 35 portmap on

2. 设定 NFS 分享出去：
[root @server root]# vi /etc/exports
/home  192.168.10.0/24(rw,async,no_root_squash)
/disk1 192.168.10.0/24(rw,async,no_root_squash)
[root @server root]# exportfs -rv
[root @server root]# /etc/rc.d/init.d/nfs start
[root @server root]# chkconfig --level 35 nfs on

3. 设定预计的挂载点：
[root @server root]# mkdir -p /cluster/node1
[root @server root]# mkdir -p /cluster/node2
[root @server root]# mkdir -p /cluster/node4
[root @server root]# mkdir -p /cluster/server
```

  - Slave：

```
1. 启动 portmap 并且设定开机启动：
[root @node1 root]# /etc/rc.d/init.d/portmap start
[root @node1 root]# chkconfig --level 35 portmap on

2. 设定 NFS 分享出去：
[root @node1 root]# vi /etc/exports
/disk1 192.168.10.0/24(rw,async,no_root_squash)
[root @node1 root]# exportfs -rv
[root @node1 root]# /etc/rc.d/init.d/nfs start
[root @node1 root]# chkconfig --level 35 nfs on

3. 设定预计的挂载点：
[root @node1 root]# mkdir -p /cluster/node1
[root @node1 root]# mkdir -p /cluster/node2
[root @node1 root]# mkdir -p /cluster/node4
[root @node1 root]# mkdir -p /cluster/server
```

挂载程序：

- Master：
  将底下这些指令测试执行一下，如果成功后，将指令写入 /etc/rc.d/rc.local 当中

```
[root @server root]# mount -t nfs -o bg,intr server.cluster:/disk1 /cluster/server
[root @server root]# mount -t nfs -o bg,intr  node1.cluster:/disk1 /cluster/node1
[root @server root]# mount -t nfs -o bg,intr  node2.cluster:/disk1 /cluster/node2
[root @server root]# mount -t nfs -o bg,intr  node4.cluster:/disk1 /cluster/node4
```

- Slave：

  将底下这些指令测试执行一下，如果成功后，将指令写入 /etc/rc.d/rc.local 当中

```
[root @node1 root]# mount -t nfs            server.cluster:/home  /home
[root @node1 root]# mount -t nfs -o bg,intr server.cluster:/disk1 /cluster/server
[root @node1 root]# mount -t nfs -o bg,intr  node1.cluster:/disk1 /cluster/node1
[root @node1 root]# mount -t nfs -o bg,intr  node2.cluster:/disk1 /cluster/node2
```

```
[root @node1 root]# mount -t nfs -o bg,intr  node4.cluster:/disk1 /cluster/node4
```

呵呵！这样就设定成功了！我们每一部主机『看起来』就好像有 400 GB 的硬盘空间啊！可怕了吧！ ^_^

- NIS 架设规划

NIS 的设定也是很简单，不过主要还是需要分为 NIS Server 与 NIS Client 两部份来设定的！请注意，在设定之前，就已经要将 NFS 搞定喔！这些流程都是有一定程度的相关性的呢！

  - Master：
    在 Master 上面需要进行的工作很多喔！首先，一定要修改 ypserv.conf 以及其它相关的档案的呐！

```
1. 启动 time 与 time-udp 这两个预先要启动的 daemon
[root @server root]# chkconfig --level 35 time on
[root @server root]# chkconfig --level 35 time-upd on
[root @server root]# /etc/rc.d/init.d/xinetd restart

2. 建立 NIS 的领域名称 (我这里是设定为 cluster )：
[root @server root]# nisdomainname cluster
[root @server root]# echo "/bin/nisdomainname cluster" >> /etc/rc.d/rc.local
[root @server root]# echo "NISDOMAIN=cluster" >> vi /etc/sysconfig/network

3. 建立 NIS 设定档：
[root @server root]# vi /etc/ypserv.conf (在这个档案内增加三行即可)
127.0.0.0/255.255.255.0    :  *  :  *  :  none
192.168.10.0/255.255.255.0:  *  :  *  :  none
*                          :  *  :  *  :  deny
[root @server root]# touch /etc/netgroup

4. 启动 NIS：
[root @server root]# /etc/rc.d/init.d/ypserv    start
[root @server root]# /etc/rc.d/init.d/yppasswdd start
[root @server root]# chkconfig --level 35 ypserv on
[root @server root]# chkconfig --level 35 yppasswdd on

5. 制作数据库：(每次有更动使用者信息时，就必须要进行这个步骤！)
```

```
[root @server root]# /usr/lib/yp/ypinit -m
[root @server root]# chkconfig --level 35 ypserv on
[root @server root]# chkconfig --level 35 yppasswdd on
```

- o
- o Slave：
  至于 NIS Client 则是需要设定 yp.conf 这个档案呢！

```
1. 建立 NIS 的领域名称 (我这里是设定为 cluster )：
[root @node1 root]# nisdomainname cluster
[root @node1 root]# echo "/bin/nisdomainname cluster" >> /etc/rc.d/rc.local
[root @node1 root]# echo "NISDOMAIN=cluster" >> vi /etc/sysconfig/network

2. 建立 NIS 查寻的主机名称：
[root @node1 root]# vi /etc/yp.conf
domain cluster
ypserver server.cluster

3. 修改密码验证方式：
[root @node1 root]# vi /etc/passwd (在这个档案的最底下新增如下一行)
+::::::
[root @node1 root]# vi /etc/nsswitch.conf
passwd:     files nis nisplus
shadow:     files nis nisplus
group:      files nis nisplus
hosts:      files nis dns

4. 启动 NIS：
[root @server root]# /etc/rc.d/init.d/ypbind    start
[root @server root]# chkconfig --level 35 ypbind on
```

- o

- 呵呵！不啰唆！马上就设定妥当啦！
- ---
- RSH 设定

  这个 RSH 已经提过了，主要的功能是提供 Master 可以使用 R 指令(如 rsh, rlogin, rcp 等等)来进行 slave 端主机的操控的！所以啦， RSH daemon 主要是在 slave 机器上面架设的喔！与 Master 就无关啦！Master 只要能够执行 R command 即可！虽然是如此，不过，在我的测试当中，最好 Master 也启动 RSH 比较好一些些啰！在底下的设定当中，我们假设 Server 上面的所有使用者都可以使用 R command 呢！设定的方法很简单啊！

- o Slave & Master：
  底下的设定在 Master 与 Slave 上面都需要动作喔！设定一样即可！

```
1. 启动 RSH 啰！
[root @node1 root]# chkconfig --level 35 rsh on
[root @node1 root]# /etc/rc.d/init.d/xinetd restart

2. 编辑可使用 R command 的主机设定文件：
[root @node1 root]# vi /etc/hosts.equiv
server.cluster +
```

- o

  特别注意，由于 RSH 预设就是不支持 root 使用 R command ，所以您必须要到 master 上面去，并以一般身份使用者进行 R command 的测试才行喔！不要直接以 root 工作，会无法成功的啦！(显示 permission deny 的啦！)

•

- o Master：
  由于 RSH Server 上面的设定中，您的 User 家目录必须要存在一个名为 .rhosts 的档案，原本我的 Server 上面就有一个名为 test 的使用者，而并且为了让我未来新建立的使用者都能够使用 R command ，所以我在 Master 这部机器上面做了这样的动作喔：

```
[root @server root]# vi /home/test/.rhosts
server.cluster
[root @server root]# vi /etc/skel/.rhosts
server.cluster
[root @server root]# chmod 644 /etc/skel/.rhosts
```

- o

- 这样就妥当的设定好了 RSH 啰！

-

- 安装 Fortran 90 的编译程序 PGI pgf90 (PS. server version)

  我由 PGI 的官方网站下载了最新的 PGI Server 套件，请特别留意的是，由于 PGI 有两种模式，一种是工作站(Workstation)一种则是服务器(Server)模式，其中，工作站仅能提供单一主机来操作，无法进行 Cluster 的功能的！因此，请务必要下载 Server 的版本，并且是支持 Linux 版本的喔！不要搞混了！ PGI Fortran Server version 的下载网址在这个地方：http://www.pgroup.com/downloads.htm，请自行下载吧！比较需要留意的是，从上面这个网站下载的版本仅是分享软件的版本，您安装之后可以具有 15 天的免费使用期限，超过期限之后，又需要重新安装一次，很是麻烦的啦！如果您的 Cluster 是用来进行学术研究的，那么在测试完成之后，可能需要去他的网站注册，这个注册的费用差异可就很大了～因为未来我的 Cluster 需要一直不断的运作，因此是需要去注册的啦！并且，我只会用到 Fortran 这个编译器，因此，我就直接使用 PGIHPF 这个版本来测试安装而已，而不是使用全部 (含 PGI Fortran 与 C) 的版本喔！因为注册的价差差了两～三万台币啊！安装 Fortran 真是很简单的啦！假设您将

linux86-HPF.tar.gz 放置在 /root/software 底下，则：(注：以下的动作仅只要在 Master 上面进行即可喔！)

```
1. 建立 pgi fortran 在 /cluster/server/program/pgi 底下：
[root @server root]# cd /usr/local/src
[root @server src]# mkdir pgi-fortran; cd pgi-fortran
[root @server pgi-fortran]# tar -zxvf /root/software/linux86-HPF.tar.gz
[root @server pgi-fortran]# ./install
接下来会有一些问题，请依序回答您的问题喔！
至于授权嘛！请建立吧！
最重要的地方，是在第三个问题，他会问你要安装的目录，请选择
/cluster/server/program/pgi

2. 修改个人参数：由于 RSH 不以 root 工作，所以我以使用者 test 来测试：
[root @serer root]# vi /home/test/.bashrc
# 加入这几行关于 PGI 的咚咚：
PGI=/cluster/server/program/pgi
export PGI
PATH=$PGI/linux86/bin:$PATH

3. 设定查寻路径：
[root @server root]# vi /etc/man.config
# 加入这一行：
MANPATH /cluster/server/program/pgi/man
```

- 这样就好了吗？！没错！确实是这样就完成了！很是简单吧！ ^_^要注意的是：
  - 记得 pgf90 必须要让所有的 node 都能够读的到，所以一定要安装在 Server 的分享出去的目录当中，我这里的例子就是安装在 /cluster/server/program/gpi 这个目录当中啰！
  - 执行档要能够执行，当然是必须要让目录在 PATH 这个变量底下，而我的 pgf90 是在 /cluster/server/program/pgi/linux86/bin 底下，所以，您的 PATH 必须要含有这个目录才行！

大概就是如此啰！

- 安装 MPICH

前面我们提过了，安装 MPICH 是平行运算里面最重要的一项工作了！因为我们就是靠他来帮我们达成运作的啊！那么怎么来安装呢？又是简单得不得了啊！首先，请先下载 mpich 吧！下载的网址在底下，我是以 mpich 1.2.5-1a 来测试的喔！

http://www-unix.mcs.anl.gov/mpi/mpich/download.html

假设您将 mpich 下载在 /root/software 里面，并且预计要安装到 /cluster/server/program/mpich 当中，而且仅安装 Fortran 而已的话，可以这样做：

```
1. 建立 mpich 在 /cluster/server/program/mpich 底下：
[root @server root]# cd /usr/local/src
[root @server src]# tar -zxvf /root/software/mpich.tar.gz
[root @server src]# cd mpich-1.2.5
[root @server mpich-1.2.5]# ./configure --enable-debug \
> -fc=pgf77 -f90=pgf90  \
> --prefix=/cluster/server/program/mpich
[root @server mpich-1.2.5]# make && make install

2. 建立可以利用的主机状态：
[root @server mpich-1.2.5]# cd /cluster/server/program/mpich/share
[root @server share]# vi machines.LINUX
node1.cluster:2
node2.cluster:2
node4.cluster:2
server.cluster:2
# 这个档案当中，格式为 <主机名称>:<主机的 CPU 个数>

3. 建立需要的变量：(又是以 test 为准喔！)
[root @server root]# vi /home/test/.bashrc
# 加入这一些数据：
PATH=$PATH:/cluster/server/program/mpich/bin
export PATH
MPI_HOME=/cluster/server/program/mpich
MPI_ARCH=$MPI_HOME/bin/tarch
export MPI_ARCH MPI_HOME
[root @server root]# vi /etc/man.config
# 加入这一行：
MANPATH /cluster/server/program/mpich/man
```

呵呵！这样就已经完成了 MPICH 的安装与设定了！就跟你说很简单吧！但是呢，要测试可就得需

要特别留意了，因为 root 预设是不许使用 RSH 的，所以测试一定要使用一般身份的使用者，这里我以 test 这个人做为测试的使用者喔！所以，请以 test 的身份登入主机，并且，这个 test 必须要在所有的主机上面都可以被查询的到才行(请参考 NIS 的设定喔！)。

```
[test @server test]$ cp -r /cluster/server/program/mpich/examples/ .
[test @server test]$ cd examples
[test @server examples]$ make pi3f90
[test @server examples]$ mpirun -np 8 pi3f90
# 上面那个 -np 后面接的就是使用 CPU 的个数啦！因为我有 8 个 node ，
# 所以当然就以最大的 CPU 个数来测试看看，如果要看到底 CPU 有没有启动的话，
# 可以先登入各个 slave 的主机，然后执行『 top -d 1 』来观察 CPU 的使用率，
# 再执行上面这个程序，就能够知道 CPU 有没有运作了！ ^_^
```

呵呵！没想到 PGI 的试用版本就能够提供多颗 CPU 的 Cluster 运作，真是给他很高兴！这样既然可以测试成功了，自然就可以去向 PGI 的官方网站注册了！注册费用不低，但是挺值得的啊！

其它主机相关设定：

除了一些基本的 Cluster 设定之外，您的主机其实可能还需要其它的设定项目的！最常见的可能就是 X-Window System 的 server/slave 架构了！您总不希望老是呆在 cluster 前面吧？！粉吵的呢！这个时候，或许就会希望可以在 Client 端连接上 Server 的 X Server 啰！

- X-Window Server/Slave 架构

架设一个很简单的 X Server 真的是很容易，容易到让你会很想笑出来～在上面的环境设定好了之后，请注意：『您的主机应该是在 run-level 为三的文字模式底下，也就是没有 X Window 存在』才对！这个时候，要设定完成 X Server ，仅要修改一个档案，并且启动一支程序即可！先说明一下我的环境：

X Server 主机的 IP 是 192.168.10.30，操作系统是 Red Hat 9；
X Client 的 IP 是 192.168.10.100 ，操作系统同时是 Linux 与 Windows 2000 。

  - X Server 的设定：
    由于我们需要启动监听来自 X Client 的要求，这个时候需要重新做一下设定：

```
1. 先备份原先的设定档：
[root @server root]# cd /etc/X11/gdm
[root @server gdm]# mv gdm.conf gdm.conf.bak

2. 编辑设定档，只要底下两行即可！
```

```
[root @server gdm]# vi gdm.conf
[xdmcp]
Enable=1

3. 启动 gdm :
[root @server gdm]# gdm
[root @server gdm]# netstat -uln
udp        0      0 0.0.0.0:177             0.0.0.0:*
# 如果有出现上面这行就对了!
[root @server gdm]# echo "/usr/bin/gdm" >> /etc/rc.d/rc.local
# 上面这行在设定开机时启动 gdm 啰!
```

-
- X Client 的设定(在 Linux 上面):
  如果您的 X Client 同样是 Linux ,那么:『在这目前这个 Client 端连接到 X Server 端执行 X 窗口』的功能,只要进行几个步骤就可以了!

```
0. 请务必要在 X Window 当中,进入 X Window 的方式有:
[root @client root]# startx
或
[root @client root]# init 5

1. 在 X Window 的画面当中,启用一个 shell ,然后输入:
[root @client root]# xhost + 192.168.10.30
192.168.10.30 being added to access control list
[root @client root]# init 3 (离开 X Window)

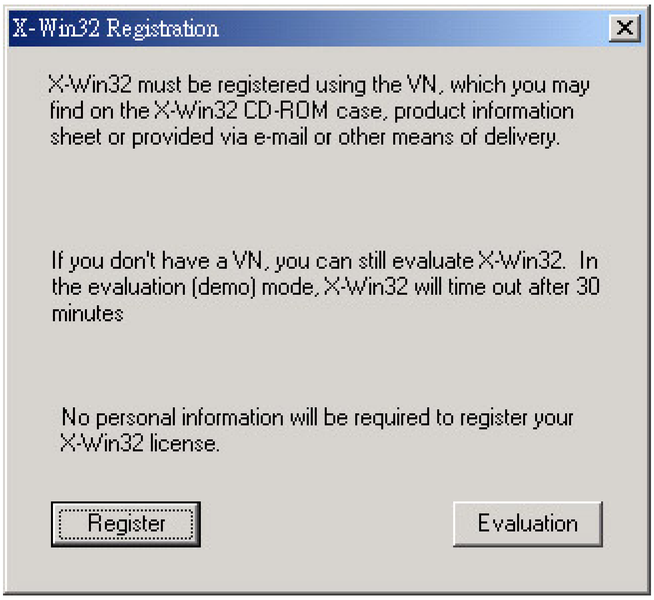
2. 在文字接口下输入:
[root @client root]# X -query 192.168.10.30
.....(进入 X Window 啰! )
```

-
- X Client 的设定(在 Windows 上面):
  如果您的 X Client 是在 Windows 上面,那么就必须要额外的来执行其它的软件了!目前您可以选择购买 Exceed 这个联机软件,或者是先『试用』 X-Win 这个软件!都是用在 Windows 上面模拟连接到 Linux X Server 用的软件啦!我这里是以 X-Win 这套程序做为介绍的,您可以在各大学的 FTP 网站上面捉到这个软件,当然,也可以在我这里下载 5.4 版。

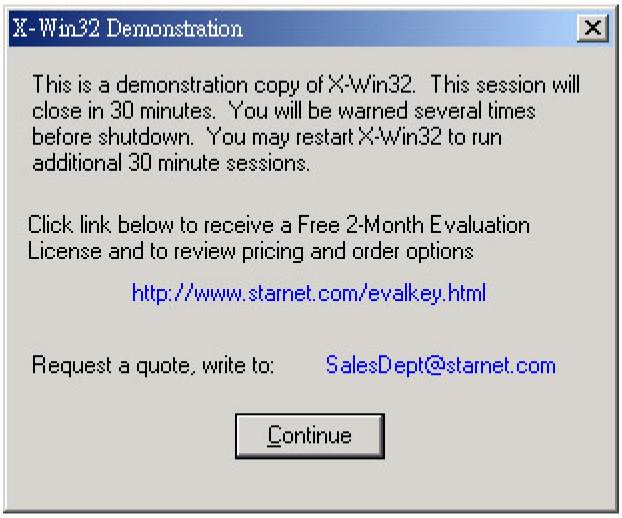
  http://linux.vbird.org/download/#x-win54

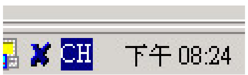
  1. 在 Windows 上的安装步骤,就是执行他即可啊!然后一直给他下一步就是了!

2. 执行 X-Win32 这个程序，在出现如下的图示后，按下 Evaluation 这个试用按钮；

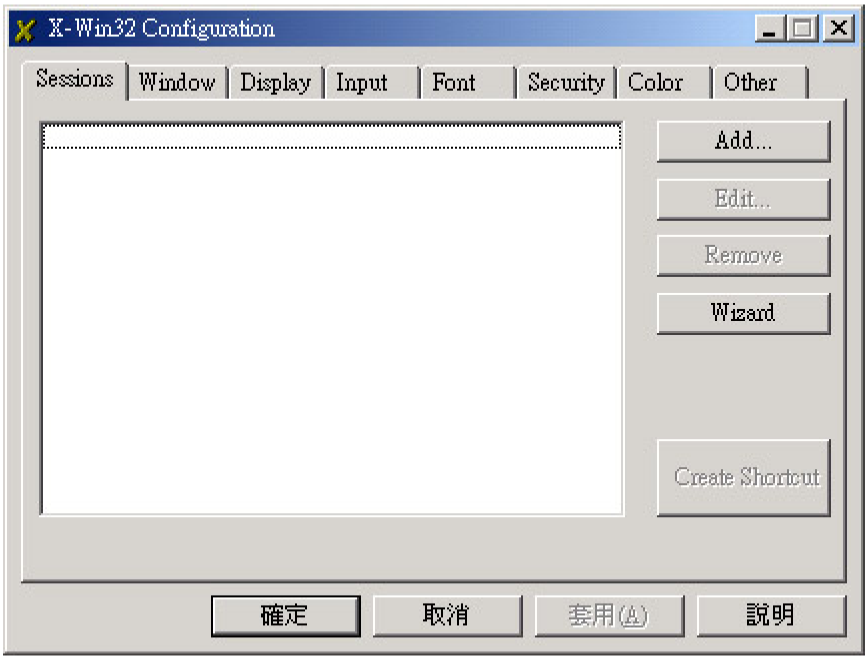


3. 按下确认按钮；

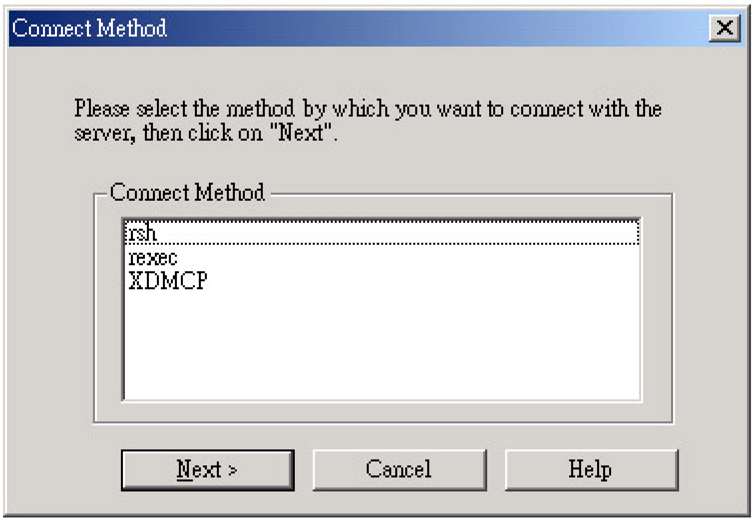


4. 在右下角的小图示当中，例如：

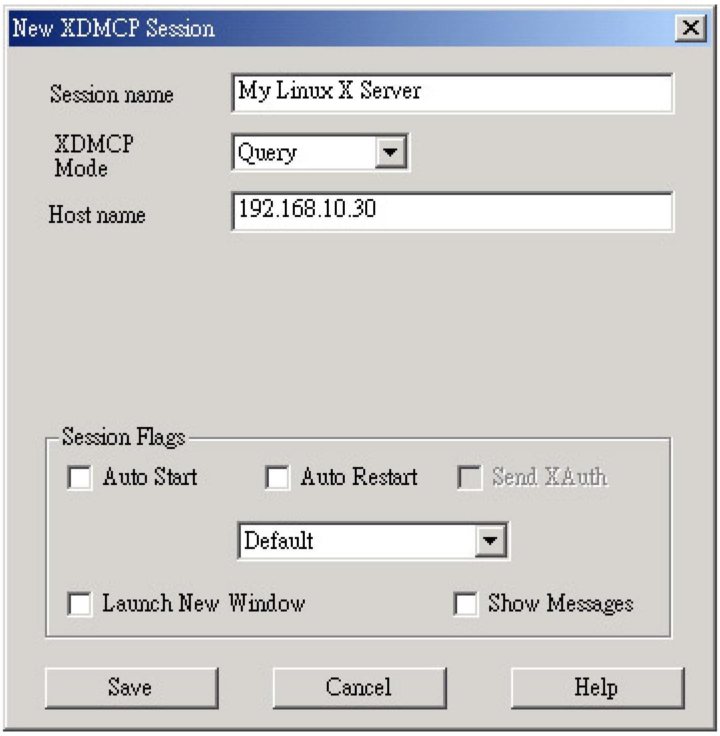
给他按下那个 X 啰，就会出现如下的图示：



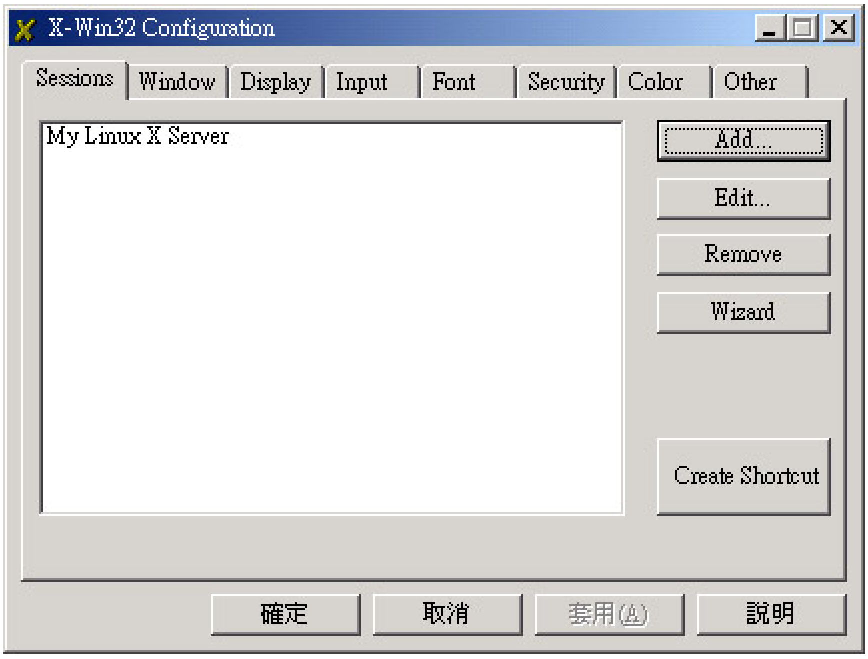
5. 按下 Add 之后，会出现如下的图示：



6.　上面选择 XDMCP 这个项目，然后选择 Next 之后，会出现：



7.　重要的地方在于 Query 这个选项，以及 192.168.10.30 这个 IP 指向喔！都设定好之后，给他 Save 一下，会出现：



8.　上面的图示当中，给他按下确定吧！然后呢？在右下角的 X 小图示中（

）按一下 X 之后，就会出现刚刚我

们建立的 My Linux X Server 这个选项，给他选择下去的啦，呵呵！就会出现啰：



很棒吧！这样就能在 Windows 上面联机进入 Linux 使用他的 X Window 功能啰！

后记：虽然架设一个 X Server/Client 是很简单的一件事，不过，如果您的机器并不是在 LAN 里面的话，而是在 Internet 上面进行 Server/Client 的 X 系统架构的连结，请特别留意的是，由于 X Window 的图形接口需要一直传输图形到您的工作机上面来，如此一来，将会损耗掉大部分的频宽喔！在我的实际案例当中，发现到我的 X Server/Client 之间流量传输达到 2000 Kbits/second, 亦即是 250KBytes/second ，还记得流量的算法吧？！如果是 ADSL 拨接架构的话，目前已经蛮常见 下载/上传 = 1.5M/386K 的传输速度，不过，即使下载达到 1.5M 了，不过在我的案例中竟然高达了 2.0M 的传输!呵呵!了解吧!所以啰,这个 X Window 的 Server/Client 架构请务必在内部网络架设就好，不要想连上 Internet 啊！会等的快睡着.....

重点回顾

- Cluster 可以是并行计算的一种，主要除了双 CPU 系统外，也可以用在多主机架构下的一种增快数值计算的方式；
- 并行计算的 Cluster 主要是藉由主从的架构去运行的，其主要的设定都在 Master 上面设定的，Slave 主要是计算功能而已，
- Cluster 上面相当重要的地方在于 NFS 的设定，因为 Master/Slaves 都需要读取同样的数据，所以 NFS 分享的档案数据就极其重要了；
- 除了 NFS 之外，还需要 MPI 这个并行计算的函式库之安装，有了 MPI ，Cluster 才能真的运作起来；
- 在 Cluster 当中，所有的主机之指令的沟通主要亦经过 RSH 的运作；

参考资源

- Marty's Linux Cluster 架设日志：http://web.csie.chu.edu.tw/~cs87668/cluster.htm
- 张裕麟先生的小文章：http://www.se.ntou.edu.tw/~ylchang/MyDocuments/MPICH-ins.txt
- 国家高速计算机中心：http://binfo.ym.edu.tw/edu/seminars/200201.files/frame.htm
- MPICH官方网站：http://www-unix.mcs.anl.gov/mpi/mpich/

架站文件习题解答篇

最近更新日期：2003/09/20

在我们的 Linux 架站文件当中，每个章节或多或少都有些课后练习给大家复习一下！呵呵！那么各个章节的解答会在这里提供喔！

PART I 、架站前进修篇：

第一章、架站之前所需的技能分析

- 请简述进行网站架设前，应该具备何种基本技能？
  基本的技能需要有：
  1. Linux 系统操作上，至少需要了解账号管理、档案属性与权限、程序与资源管理、硬件如硬盘之挂载与软件套件之安装如 RPM 套件管理员等等，而 vi 与 Shell 亦是不可忽略之基本技能，更重要者，日志的管理以及系统服务的原理(如 stand alone 与 super deamon 服务启动的方式差异等)亦需同时厘清，以方便未来架设网站时除错的技巧；
  2. 在网络的基础知识上，TCP/IP 的观念以及路由的概念相当的重要，此外，DNS 的概念也很重要。
  3. 在心态的调整上面，系统管理员需要的道德感以及使命感需要较高的标准。

- 如果我有一颗硬盘在 A 主机上面安装了 Linux 之后，拿到另一台配备相同的 B 主机上面去进行开机，结果竟然无法顺利开机，您认为可能的原因是什么？
  由于配备相同，所以排除硬件的问题，不过，考虑到 IDE 排线与 partition 的代号，以及 /etc/fstab 的对应，所以应该是由于 IDE 插槽放置错误所致(Linux 系统下，每个 IDE 插槽对应的 partition 名称皆不相同喔！)。此外，上次不正常关机也可能造成硬盘损害而无法开机成功！

- 一般来说，在 Linux 系统上，使用者预设的家目录在那个目录下？另外，新增一个使用者时，该使用者预设的家目录内容来自那个目录下？
  在 /etc/default/useradd 这个档案里面会规范使用者的预设家目录以及预设家目录的内容，一般来说，使用者预设家目录在 /home ，至于家目录内的档案则复制来源在 /etc/skel 里面。

- 磁盘配额 (quota) 能否针对某个特定的目录进行限制？Quota 有什么较为特殊的使用限制？
  Quota 目前仅能针对整个 partition 进行限额配置，如果该特定目录是一个 partition 那就可以进行 quota 设定，否则无法针对特定目录！此外， Quota 除了特定目录以及需要 ext2, ext3 等磁盘格式才支持之外，也需要核心的支持喔！

- 在 Linux 系统下，要寻找一个档名为 vbird.document 的档案，可以使用什么指令进行搜寻？又，如果要寻找在一天内更动过的档案，又该如何进行？
  如果是执行档可以使用 which command 来搜寻，如果是档案，就必须要使用 locate vbird.document 或 whereis vbird.document 或 find / -name vbird.document 。如果要找到

一天内更动过的档案，可以使用 find 配合参数，如：『 find / -ctime 1』。

- 在 Linux 系统中，常见的套件管理员有 RPM 与 Tarball ，请分别说明这两个套件管理员的优缺点。
    - RPM 套件管理员在安装套件的时候，会将该套件的文件名称、套件功能与讯息等等信息记录于 /var/lib/rpm 目录内，由于有这些套件的资本信息，因此在系统内很容易进行 RPM 的升级、安装、移除等动作。不过，由于 RPM 档案之间的相依性相当的强，因此常常会有版本不合或者是欠缺某样前驱套件的问题发生。
    - 至于 Tarball 则是原始码，使用者可以自行设定套件的编译参数，以符合自己的 Linux 平台。此外，由于 Tarball 是原始码，因此需要在您的系统上面进行编译，编译的过程中需要 gcc, make, 以及 kernel source 等套件，还有您所想要安装的套件所需要的前驱套件也同时需要先安装后，才能进行 Tarball 的安装。注意，几乎每个 Tarball 以 tar 程序解开后，在新增的目录下均会有 README 以及 INSTALL 档案，请务必先查阅过后才进行编译工作。

- 如果我下载了一个档名为 httpd-2.0.52.tar.gz 的档案，一般来说，这个档案代表的意义为何？我该如何让这个档案能够在我的 Linux 系统上面安装？
  由于附档名是 .tar.gz 或者是 .tgz 的档案，所以可以认定该档案为一个 Tarball 的档案。至于文件名的配置方面，httpd 为套件名称，2.0.52 则通常为该套件的版本名称了！那如何安装？由于该档案为 httpd 这个套件，且格式为 Tarball ，所以您必须要：
    0. 系统上面务必具有 tar, make, gcc 等相关的编译套件；
    1. 使用 tar 解开 httpd-2.0.52.tar.gz 之后，务必进入该目录内读取 REAME 或/与 INSTALL 档案，以了解是否还需要其它的相关套件的搭配安装；
    2. 以 ./configure --help 查阅一下是否有相关可以加入或者取消的编译内容；
    3. 使用 make 读取 Makefile 来编译程序；
    4. 使用 make install 来安装程序！

- 我以原始码的方式进行一个套件的安装，但是在分析系统的时候，分析程序一直告诉我找不到 cc 这个指令，请问这是什么问题？为何需要 cc ？又，我该如何解决这个问题，好让套件可以顺利的被安装在我的 Linux 上面？
  如前面几个题目所说的，因为是原始码，所以还需要编译程序来将该原始码编译成为可以在您的 Linux 系统上面跑的 binary 档案，在 Linux 上头预设的编译程序就是 gcc 这个编译器(compiler)。如果您在安装 Linux 的时候，使用 Linux Installer 预设的套件选择，那通常会没有安装 gcc 以及 make 等套件，此时，请拿出您的原版光盘，以 mount 指令挂载后，使用 RPM 将一个一个相关的套件安装即可(过程会蛮复杂的！)^_^

- 我发现我的 Linux 系统怪怪的，似乎有什么不知名的程序在内存当中跑，我该如何将这个不知名的程序捉出来，并且将他移除？
  如果要捉出程序(process)的话，可以使用 ps -aux 或者是直接输入 top 来查询 process 的 ID (PID)，找到 PID 号码后，再以 kill -9 PID 来删除该程序即可。

- 我总是无法编辑某个档案，您认为应该是什么问题造成的？那又要怎么解决？
  无法编辑某个档案，可以先使用 file 这个指令来查询一下该档案的格式，例如想察看 /etc/shadow 的格式，可以下达：『file /etc/shadow』，如果是文字文件，却还是无法编辑，那么最可能发生的原因就是『权限』的问题了。可以使用 ls -l filename 察看档案权限，再以 chmod 或 chown 来修订该档案的权限。此外，该档案也可能含有隐藏属性，可以使用 lsattr filename 查阅，再以 chattr 来修订隐藏属性。

- 什么是 UID 与 GID ？UID 有哪些等级？
  在 Linux 系统下，使用者与群组其实都是以『ID(数字)』的格式来设定的，所以使用者与群组其实都是 UID 或 GID (User ID 与 Group ID)，Linux 对于档案权限也都是使用 UID/GID 来分辨。不过人类习惯使用文字来记忆，所以才会有 /etc/passwd 与 /etc/group 来转译 ID 与 User 及 Group 之间，这也才会发生为何在 Internet 上面捉下来的 Tarball 解开之后，往往会有档案拥有人与群组为数字的型态，因为您 Linux 系统上面的 /etc/passwd 与 /etc/group 没有相关的对应文字说。至于 UID 的等级主要有两种，分别是超级管理员(root)，其 UID 为 0，其它非为 0 的 UID 基本上身份是相同的！不过 Linux 通常会将小于 500 的 UID 保留给系统使用。

- 使用者的家目录参数、UID、GID 以及其它相关参数，还有密码档案，放置在哪些档案里面？
  放置在 /etc/passwd 与 /etc/shadow 当中。当然还有 /etc/login.defs 喔！

- 你认为一个称职的网管人员应该具备什么能力？
  能力需求相当高，如了(1)操作系统的基础知识(不论是 Linux/Unix/MAC/MS)；(2)网络基础的知识；(3)个别 Internet Services 的运作知识之外，还需要(4)身心保持在备战状态，以及(5)具有相当高程度的道德感、责任感与使命感。

- 我要启动一个系统预设的 Service，请问我可能可以由执行或修改哪些目录底下的档案来启动？
  如果是 stand alone 的服务，可以经由 /etc/rc.d/init.d/* 里面的档案，如 /etc/rc.d/init.d/syslog start；
  如果是 Super daemon 的服务，就必须(1)先到 /etc/xinetd.d 或者是 /etc/inetd.conf 修改相关档案或参数；(2)以 /etc/rc.d/init.d/xinetd restart 来启动。

- 我要关掉 cron 这个服务，应该怎么关掉他？如果正常的方法无法关闭这个服务，可以使用什么方法来关闭？
  因为 cron 是一个 stand alone 的服务，所以可以使用 /etc/rc.d/init.d/cron stop 来关闭；如果还是无法正常关闭，可以使用 ps -aux | grep cron 捉出该程序的 PID ，然后以 kill -9 PID 来关闭。

- 如果一开机就要执行某个程序，应该要将该程序写入那个档案里面？
  可以直接在 /etc/rc.d/rc[run-level].d 里面加入 S 开头的档案，不过，更简单的作法是直接将该程序写入 /etc/rc.d/rc.local ，不过，请注意该程序必须要具有可执行的权限，且 rc.local 也必须要是可执行喔！

第二章、简易网络基础架构

- 请简述 OSI 网络七层协议每一层的功能;
  OSI 网络七层协议主要又分为两大部分，网络层与使用者应用层两部份。至于每一层的相关功能请参考本章节的表一所示。

- 在 ISP 提供的网络服务中，他们提到传输速度为 1.5M/382K ，请问这个数据的单位为何?
  由于电子讯号的基本单位为 bits ，所以一般 ISP 提供的传输速度单位均为 bits/second ，并非我们常用的档案计数单位 Bytes ！

- 什么是 MAC (Media Access Control) ，MAC 主要的功能是什么?
  在网络媒体上面，数据要传输时，必须知道下一个节点的地址才能顺利传送，这个节点的地址如网络卡的硬件地址就是 MAC 了。硬件地址是在网络卡出厂时就已经焊死在上面了(某些笔记型计算机的 MAC 则可以经由特殊软件修改)，而软件地址，则是我们常说的 IP ，这两个并不相同喔!其中，MAC 与 IP 互相的对应则是以 ARP Table 来进行转译的!

- 请解释什么是物理广播 (Physical Broadcast) ，他与逻辑广播 (Logical Broadcast) 有什么不同?
  物理广播主要与网络媒体有关，在同一个网络媒体上面同一时间仅能有一部主机来使用这个网络共享媒体，而要判断目前有没有其它主机在使用这个网络媒体时，就是利用 Physical Broadcast 咯!
  至于逻辑广播呢？呵呵！就是在查询到底网域上面有没有我们这部主机想要连接的相关协议或者同一网域内的其它主机了！他则主要与 MAC, ARP 以及软件地址(通称的 IP)有关啦!

- 什么是封包碰撞？为什么会发生封包碰撞?
  当两部主机同时在一个网络媒体上面进行数据传输时，两个数据封包就会发生碰撞的情况，这就是封包碰撞了。在网络媒体流量很高、网络媒体的联机长度过长都会容易发生数据封包碰撞的情况。

- ARP Table 的作用为何？如何在我的 Linux 察看我的 ARP 表格?
  ARP Tables 主要在对应 IP 与 MAC ，当主机要将数据封包送出到下个节点时，必须要知道下个节点的 MAC 才能传送，而如果不知道 MAC 时，就得需要使用逻辑广播来查询 IP 与 MAC 的对应才行。不过，当您的主机内部的 ARP table 已经记录了 IP 对应的 MAC 之后，那么该资料封包则可以立即传送到下个节点去，而不需要再次的进行逻辑广播了。要知道目前的 arp table 内有多少纪录，可以使用 『 arp -a 』来查询。

- 简略说明 Netmask 的作用与优点;
  Netmask 可以有效的增加网络的效率，这是因为 Netmask 可以定义出一个网域的大小，那么 broadcast 的时间就可以降低很多！一般来说，我们如果要将一个大网域再细分为小网域，也需要藉由 Netmask 来进行 subnet 的切割。

- 我有一组网域为：192.168.0.0/28 ，请问这个网域的 Network, Netmask, Broadcast 各为多少?
  而可以使用的 IP 数量与范围各是多少?

因为共有 28 个 bits 是不可动的，所以 Netmask 地址的最后一个数字为 11110000，也就是 (128+64+32+16=240) ，所以：
Network：192.168.0.0
Netmask：255.255.255.240
Broadcast：192.168.0.15
IP：由 192.168.0.1 ~ 192.168.0.14 共 14 个可用 IP 喔！

- 承上题，如果网域是 192.168.0.128/29 呢？
因为是 29 个 bits 不可动，所以最后一个 Netmask 的地址为： 11111000 也就是 (128+64+32+16+8=248)，所以：
Network：192.168.0.128
Netmask：255.255.255.248
Broadcast：192.168.0.135
IP：由 192.168.0.129 ~ 192.168.0.134 共 6 个可用的 IP 喔！

- 我要将 192.168.100.0/24 这个 C Class 的网域分为 4 个子网域，请问这四个子网域要如何表示？
既然要分为四个网域，也就是还需要藉助 Netmask 的两个 bits (2 的 2 次方为 4 啊！)，所以 Netmask 会变成 255.255.255.192 ，每个子网域会有 256/4=64 个 IP ，而必须要扣除 Network 与 Broadcast ，所以每个子网域会有 62 个可用 IP 喔！因此，四个子网域的表示方法为：192.168.100.0/26, 192.168.100.64/26, 192.168.100.128/26, 192.168.100.192/26。

- 如何观察 Linux 主机上面的路由信息 (route table)？
路由信息的观察可以下达 route 来直接察看！或者是下达 route -n 亦可！

- TCP 封包上面的 SYN 与 ACK 标志代表的意义为何？
SYN 代表该封包为该系列联机的第一个封包，亦即是主动联机的意思；
ACK 则代表该封包为确认封包，亦即是回应封包！

- 什么是三向交握？在哪一种封包格式上面才会有三向交握？
使用 TCP 封包才会有三向交握。TCP 封包的三向交握是一个确认封包正确性的重要步骤，通过 SYN, SYN/ACK, ACK 三个封包的确认无误后，才能够建立联机。至于 UDP 封包则没有三向交握喔！

第三章、局域网络架构简介

- 幻想自己是一个私人公司的老板，员工有 20 人，如果我想要让公司的员工都可以连上 Internet ，并且控管每个员工的对外联机，请问我的硬件联机应该怎么配置比较好？另外，应该申请的线路频宽应该多大较佳？而我的内部网域 (IP, Netmask, ...) 又该如何设定比较好？
因为只有 20 个员工，事实上，人员并不多，所以并不需要在内部加设 router 来隔开不同的网域。不过，由于想要管理员工的上网功能，因此最好以第三章图三的架构，以一部主机内含两块网络卡隔开 Intranet 与 Internet 两个网段，便于管理；至于对外联机的频宽方面，如果 Linux

主机需要进行网络服务的功能，那么最好能够有 512/512 的频宽，因为可能会有 Client 端下载的问题，至于如果单纯的仅下载时(公司没有额外的网络服务)，可以考虑使用 T1/384 之类的频宽，比较省钱；而在内部网域设定上，如同本章节使用的网域 192.168.0.0/24 来设定即可，如果公司内部尚有移动装置(NoteBook)，则可以在 Linux 上加装 DHCP (参考十六章)服务器。

- 承上题，假如我的私有网域内有较大的网络流量，那么我的网络媒体应该怎么选择较佳？
  如果私有网域内的流量较大，那么网络媒体需要选择 Switch 而不要使用 Hub ！另外，整体包括网络线也都需要使用能够负载 10/100 Mbps 的流量才好。

第四章、连上 Internet

- 我要如何确定我在 Linux 系统上面的网络卡已经被 Linux 捉到并且驱动了？
  网络卡能不能被捉到可以使用『 dmesg|grep eth 』来判断，有没有驱动则可以使用 lsmod 看看模块有没有加载核心！最后，以 ifconfig eth0 192.168.0.10 测试看看！

- 假设我的网络参数为：IP 192.168.100.100, Netmask 255.255.255.0, 请问我要如何在 Linux 上面设定好这些网络参数 (未提及的网络参数请自行定义！)？请使用手动与档案设定方法分别说明。
  - 手动设定为：『 ifconfig eth0 192.168.100.100 netmask 255.255.255.0 up 』
  - 档案设定为：vi /etc/sysconfig/network-scripts/ifcfg-eth0 ，内容为：

    ```
    DEVICE=eth0
    ONBOOT=yes
    BOOTPROTO=static
    IPADDR=192.168.100.100
    NETMASK=255.255.255.0
    NETWORK=192.168.100.0
    BROADCAST=192.168.100.255
    ```

    要启动则使用 ifup eth0 即可！

-
- 我要将我的 Linux 主机名称改名字，步骤应该如何(更改那个档案？如何启用？)？
  Linux 主机名称在 /etc/sysconfig/network 这个档案里面的『HOSTNAME=主机名称』来设定，先以 vi 来修改，改完后可以使用 /etc/rc.d/init.d/network restart 或者直接 reboot 启动主机名称！

- /etc/resolv.conf 与 /etc/hosts 的功能为何？
  以主机名称寻找 IP 的方法， /etc/resolv.conf 内填写 DNS 主机名称，至于 /etc/hosts 则直接填写主机名称对应的 IP 即可！

- 我使用 ADSL 拨接连上 Internet ，请问拨接成功之后，我的 Linux 上面会有几个网络接口 (假设我只有一个网络卡)？
  因为拨接是使用 PPP (点对点)协议，所以拨接成功后会多出一个 ppp0 的接口，此外，系统原本即有 eth0 及 lo 这两个界面，所以共有三个界面。

- 在 Linux 上面进行 ADSL 拨接应该使用什么软件？
  请爱用 rp-pppoe ，官方网站： http://www.roaringpenguin.com/pppoe/

- 一般来说，如果我拨接成功，也取得了 ppp0 这个接口，但是却无法对外联机成功，您认为应该是哪里出了问题？该如何解决？
  因为拨接成功了，表示物理对外联机没有问题，那么可能的问题应该是发生在 Gateway 上面了！确认的方法请使用 route -n 查阅路由信息，然后修订
  /etc/sysconfig/network-scripts/ifcfg-eth0 吧！

第五章、Linux 常用网络指令介绍

- 我要增加一个路由规则，以 eth0 连接 192.168.100.0/24 这个网域，应该如何下达指令？
  以手动的方法为：『route add -net 192.168.100.0 netmask 255.255.255.0 dev eth0』即可！

- 我的网络停顿的很厉害，尤其是连接到 tw.yahoo.com 的时候，那么我应该如何检查那个环节出了问题？
  使用 traceroute 寻找到底那个环节出问题：『traceroute tw.yahoo.com』即可！

- 我发现我的 Linux 主机上面有个联机很怪异，想要将他断线，应该如何进行？
  以 root 的身份进行『netstat -anp |more』查出该联机的 PID，然后以『 kill -9 PID 』踢掉该联机。

- 您如何知道 green.ev.ncku.edu.tw 这部主机的 IP ？
  方法很多，可以利用 host green.ev.ncku.edu.tw 或 dig green.ev.ncku.edu.tw 或 nslookup green.ev.ncku.edu.tw 等方法找出！

- 请找出您的机器上面最适当的 MTU 应该是多少？
  请利用『ping -c 3 -M do -s MTU yourIP 』找出您的 IP 的 MTU 数值。

- 如何在终端机接口上面进行 WWW 浏览？又该如何下载 WWW 上面提供的档案？
  要浏览可以使用 lynx ，至于要下载则使用 wget 这个软件。如果想要在终端机上面看到中文，还可以安装 JCMME 。

- 在终端机接口中，如何连接 bbs.sayya.org 这个 BBS ？
  利用 telnet bbs.sayya.org 即可连接上，同样的，要看到中文，还是得安装 JCMME 。

第七章、限制 Linux 对外联机的埠口

- 如何观察您 Linux 主机上面已经有多少 port 被打开了？
  如果仅想单纯了解正在 LISTEN 当中的埠口，可以使用『netstat -tul』，如果还想知道有多少联机已经建立，可以使用『netstat -an』来察看。

- 如何观察程序？
  利用『ps -aux』，或者是 top 来察看均可。

- 请问 LISTEN 的 port 与 daemon 的关系为何？
  正在 LISTEN 当中的埠口均是由某些服务(daemons)所启动的，所以要启动埠口就得启用某个服务，要了解某个埠口是由那个 daemon 所启动的，就利用 netstat -tulp 来查阅。

- 请解释三向交握的原理与封包传输的方向。
  三向交握为较为可靠的封包传输的一种确认方式，因此只有 TCP 封包才能具有三向交握。他利用(1) client 对 Server 主动联机时带有的 SYN 标志，(2)Server 响应时的 SYN/ACK 及(3)最终 Client 确认的 ACK 标志来确认封包的可靠性。更详细说明请参考第二章 TCP 与三向交握。

- 请问 stand alone 与 super daemon 各是什么？
  Linux 系统的服务有独立启动(stand alone)及超级服务员(super daemon)两种启动的方式。挂在 super daemon 底下的服务可以经由 super daemon 的控管，以加强一些安全功能，不过由于还要经过 super daemon 的管理，所以服务的连接速度上会比 stand alone 慢一点。详细的说明请参考『鸟哥的 Linux 私房菜 -- 基础学习篇』第二十一章『认识系统服务』内的相关说明吧！

- 请问您的 Linux 主机 (不论是那个 distributions) 有关 daemon 启动与关闭的 scripts 与档案放置在那个目录下？
  各个 daemons 的启动与关闭的 scripts 是放置在 /etc/rc.d/init.d 里面，至于 super daemon 的控管参数档案则在 /etc/xinetd.d 里面！

- 请将您的 linux 主机对外的联机端口口全部关闭！
  请参考本章节的作法！利用 ntsysv 或者 chkconfig 等功能加上 reboot ，或者 netstat 配合 kill 的方式！

第八章、Linux 网络套件升级

- 请依照您的 Linux 系统进行适合的网络套件升级程序(一个步骤一个步骤写下来)，并说明为何您需要选择这样的网络升级程序？
  这题请参考本章节的内容，选择 APT 或者是其它 Linux distributions 网站提供的在线升级方

式来进行您的套件升级。

第九章、多 IP 与 Router 的架设

- 请问您如何将您的 eth0 这个接口修改成为 192.168.100.2 在网域 192.168.100.0/25 之内的网络参数内容？
  因为 192.168.100.0/25 的 netmask 为 255.255.255.128 ，所以可以这样做：『ifconfig eth0 192.168.100.2 netmask 255.255.255.128 up 』这样即可！如果尚须其它的参数，则需要以档案形式来下达，如 vi /etc/sysconfig/network-scripts/ifcfg-eth0，并修改为：

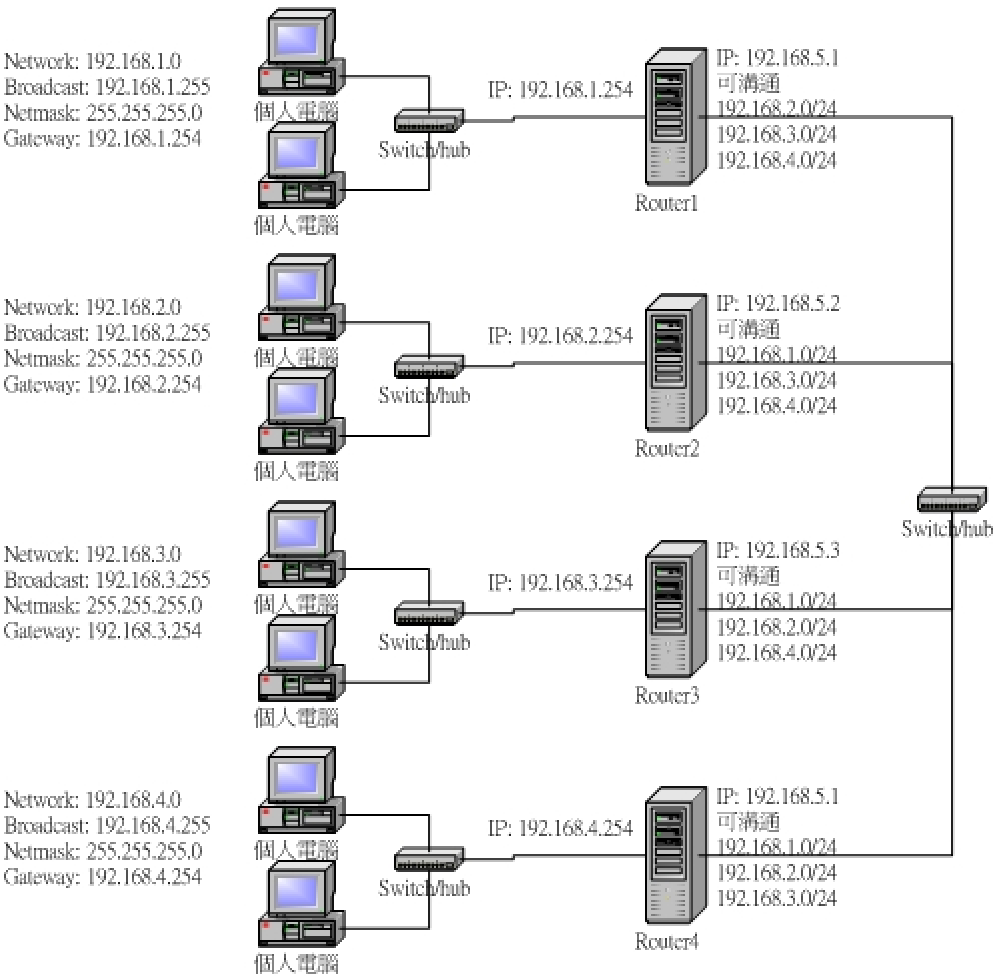
```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.100.2
NETMASK=255.255.255.128
NETWORK=192.168.100.0
BROADCAST=192.168.100.127
```

- 请手动设定 eth0:1 这个虚拟接口，使成为网络参数： 192.168.200.2, 网域在 192.168.200.0/24。
  ifconfig eth0:1 192.168.200.2 up

- 如何观察路由表？
  route -n 即可查阅！注意到 0.0.0.0 那个目标(default gateway)。

- 如何启动 Linux 的 IP Forward 功能？
  直接以『echo "1" > /proc/sys/net/ipv4/ip_forward 』即可！

- 假设您是一个学校单位的信息管理员，学校内有 200 部计算机，奉上面大头的旨意，必须要将 200 部计算机分为 4 个 Subnet ，请问您应该如何布线(请画出示意图)？而这 4 个 Subnet 的网络参数如何选择(请自行选择)？而是否需要 Router ？如果需要的话，假设每个 Router 仅能有两个网络实体接口，那么该如何布线？(注：不要使用虚拟接口)
  我的布线如同下图所示：



每个 Router 都具有两个界面，且四个 Router 的右边界面都在同一个网段内！那么 Router 1 怎么跟 Router 2 的内部网域进行沟通？利用：『route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.5.2』即可！

- 万一您的网络有点停顿，发现可能是网络上某个节点出现问题，您应该如何确认是哪一部 Router 出问题？
  使用第五章 Linux 常用网络指令当中提到的 traceroute 来查询！

第十章、认识网络安全

- 我老是发现我的系统怪怪的，似乎有点停顿的模样，怀疑可能是 CPU 负荷太大，所以要去检查一下系统相关的信息。请问，我该以什么指令去检查我的系统相关的信息？
  可以使用 top, sar, free, ps -aux, uptime, last 等功能去查询系统的相关信息喔！

- 我怀疑我的系统上面有过多的具有 SUID 的档案存在，导致一般使用者可以随意的取得 root 的权限，请问，我要如何找出这些具有 SUID 权限的档案？
  因为 SUID 是 4000 这个权限的模样，所以我可以这样做：

find / -perm +4000

- 我由国内一些 ftp 网站上下载了 Red Hat 公司释出的套件，我想安装他，但又不知道该套件档案是否被修改过！请问我该如何确定这个套件的可用性？
  利用最简易的 MD5 编码来测试一下，例如『 md5sum 套件名称』，再比对与原始套件释出的 MD5 数据是否相同！？

- 良好的密码规划是防备主机的第一要务，请问 Linux 系统当中，关于密码相关的档案与规则设定在哪些档案里面？
  密码的设定规则在 /etc/login.defs 里面！至于密码档案在 /etc/shadow 内！

- 简易说明，当一部主机被入侵之后，应该如何处理？
  找出问题、重新安装、漏洞修补、数据还原！请参考本章最后面的『修补工 0240network-secure-1.php#repaire 作』内容！

第十一章、简易 Firewall 架设

- 为什么我架设了防火墙，我的主机还是可能中毒？
  防火墙不是万灵丹，他还是可能被病毒或者是木马程序所入侵的！此外，如果您的主机本身就已经提供了多个网络服务，则当该网络服务的套件有漏洞时，防火墙仍然无法克服该服务的漏洞的！因此仍然需要持续的进行主机的监视工作！

- 请说明为何架设了防火墙，我的主机还是可能被入侵？入侵的依据可能是什么方法？
  因为防火墙仅是抵挡某些不受欢迎的封包，如果您有开放 WWW 的服务时，则要求您主机 port 80 的封包将可直接进入您的主机，万一 WWW 套件有漏洞时，那么就可能被入侵了！所以套件的更新很重要！

- 我们知道核心为 2.4 的 Linux 使用的防火墙机制为 iptables ，请问，如何知道我的 Linux 核心版本？
  利用 uname -r 可以查得！

- 请列出 iptables 预设的两个 table ，以及各个 table 里面的 chains 与各个 chains 所代表的意义；
  filter 为预设的 Table，里头预设的链有：
    - INPUT：为来自外部，想要进入主机的封包；
    - OUTPUT：为来自主机，想要离开主机的封包；
    - FORWARD：为主机内部网域与外部网域的封包(不论进或者出)，但该封包不会进入主机。

  还有 nat 这个 table：

    - PREROUTING：进行路由之前的封包传送过程

- OUTPUT：离开主机的封包传送过程；
- POSTROUTING：已经经过路由了，然后才进行的过滤规则。

- 什么是 iptables 的预设政策 (Policy)？
  当封包的所有属性都不在防火墙的规则当中时，那么这个封包能否顺利的通过防火墙，则以 Policy 作为这个封包的最终动作了！

- 假设今天我的 Linux 仅是作为 Client 之用，并没有对 Internet 进行任何服务，那么您的防火墙规划应该如何设定比较好？！
  既然没有对 Internet 提供任何服务，那么(1)请将所有的对外埠口先关闭吧！(2)防火墙规则当中，最重要的是 INPUT 的 Policy 一定要 DROP ，然后将『 iptables -A INPUT -m state --state RELATED -j ACCEPT 』即可！

- 我要将来自 192.168.1.50 这个 IP 来源的封包，只要是向我的 21~23 埠口要求的封包，就将他抵挡，应该如何下达 iptables 指令？
  iptables -A INPUT -p tcp -s 192.168.1.50 --dport 21:23 -j DROP

- 我要将我自己主机 ping 的响应功能取消，应该如何下达 iptables 的指令？
  因为 ping 能否响应用的是 icmp 的 type 8 (请参考第二章 网络基础 内的 ICMP 相关内容)，所以我可以这样做：
  iptables -I INPUT -p icmp --icmp-type 8 -j DROP

- 请说明为何这个指令是错误的？『iptables -A INPUT -p udp --syn -s 192.168.0.20 -j DROP』？
  因为只有 TCP 封包才会具有 SYN 的标志， UDP 并没有 SYN 的标志啊！所以上面的指令是错误的！

- DNS 的要求是必须的，那么我该如何设定我的主机可以接受要求 DNS 的响应呢？
  因为 DNS 的来源是 port 53 ，因此要接受来自 port 53 的封包就成为了：
  iptables -A INPUT -p udp --sport 53 -j ACCEPT
  iptables -A INPUT -p tcp --sport 53 -j ACCEPT

- 如何取消 iptables 在我的系统上面？
  先要清除规则后，才能够将 iptables 移除！不过，我们主要将规则清除即可！
  iptables -F; iptables -X; iptables -Z
  iptables -t nat -F; iptables -t nat -X; iptables -t nat -Z

- 如何储存目前的防火墙机制，以及如何将上次储存下来的机制回复到目前的系统中？
  请利用 iptables-save 以及 iptables-restore 这两个指令，配合命令重导向即可！

第十二章、申请合法的主机名称

- 请简易说明 /etc/hosts 的用途；
  这个档案是早期用在进行主机名称与 IP 的解析的，目前比较常用在内部网域的名称解析上，可以加快内部网域的反查喔！

- 请说明『合法授权』的主机名称需要做什么？
  如果想要合法授权，就需要向上层 DNS 主机『注册』才行！而且还要上层 DNS 主机管理员愿意将领域名称的解析权限授权给您啊！

- 什么是动态 DNS 系统？(仅说明 client 端)
  因为我们的 Client 拨接时，得到的 IP 都不是固定的，所以无法以 DNS 系统进行固定 IP 对应主机名称的工作!此时就需要动态 DNS 系统了!以 DNS 主机提供的动态更新主机名称对应 IP 的机制，可以让我们的不同 IP 对应到同一个主机名称呐！

- 如果您使用 adsl 拨接来上网设定服务器，那么该申请哪一类型的主机名称？为什么？
  因为我是以 ADSL 上网拨接，所以 IP 是不固定的，此时需要申请动态 DNS 主机的主机名称，例如 adsldns.org 以及 no-ip.org 等等！

PART III、各类服务器架设篇：

第十三章、简易 Telnet 与 SSH 主机设定

- Telnet 与 SSH 都是远程联机服务器，为何我们都会推荐使用 SSH 而避免使用 Telnet 呢？原因何在？
  因为 Telnet 除了使用『明码』传送数据外，本身 telnet 就是很容易被入侵的一个服务器，所以当然也就比较危险了。至于 ssh 其实也不是很安全的！由台湾计算机危机处理小组的文件可以明显的发现 openssl + openssh 也是常常有漏洞在发布！不过，比起 telnet 来说，确实是稍微安全一些！

- 请尝试说明 SSH 在 Server 与 Client 端联机时的封包加密机制；
  利用 key pair 来达到加密的机制：Server 提供 Public Key 给 Client 端演算 Private key ，以提供封包传送时的加密、解密！

- 请问 SSH 的设定档是哪一个？如果我要修改让 root 无法使用 SSH 联机进入我的 SSH 主机，应该如何设定？又，如果要让 badbird 这个使用者无法登入 SSH 主机，该如何设定？
  SSH 设定档档名为 sshd_config ，通常放置在 /etc/ssh/sshd_config 内；如果不想让 root 登入，可以修改 sshd_config 内的参数成为：『PermitRootLogin no 』，并重新启动 ssh 来设定！如果要让 badbird 使用者无法登入，同样在 sshd_config 里面设定为：『DenyUsers badbird』即可！

- 在 Linux 上，预设的 Telnet 与 SSH 服务器使用的埠口(port number)各为多少？
  telnet 与 ssh 的埠口分别是：23 与 22！请参考 /etc/services 喔！

- 如果发现我无法在 Client 端使用 ssh 程序登入我的 Linux 主机，但是 Linux 主机却一切正常，可能的原因为何？(防火墙、known_hosts...)
  无法登入的原因可能有很多，最好先查询一下 /var/log/messages 里面的错误讯息来判断，当然，还有其它可能的原因为：
  1. 被防火墙挡住了，请以 iptables -L -n 来察看，当然也要察看 /etc/hosts.deny；
  2. 可能由于主机重新开机过， public key 改变了，请修改您的 ~/ssh/known_hosts 里面的主机 IP ；
  3. 可能由于 /etc/ssh/sshd_config 里面的设定问题，导致您这个使用者无法使用；
  4. 在 /etc/passwd 里面，您的 user 不具有可以登入的 shell ；
  5. 其它因素(如账号密码过期等等)

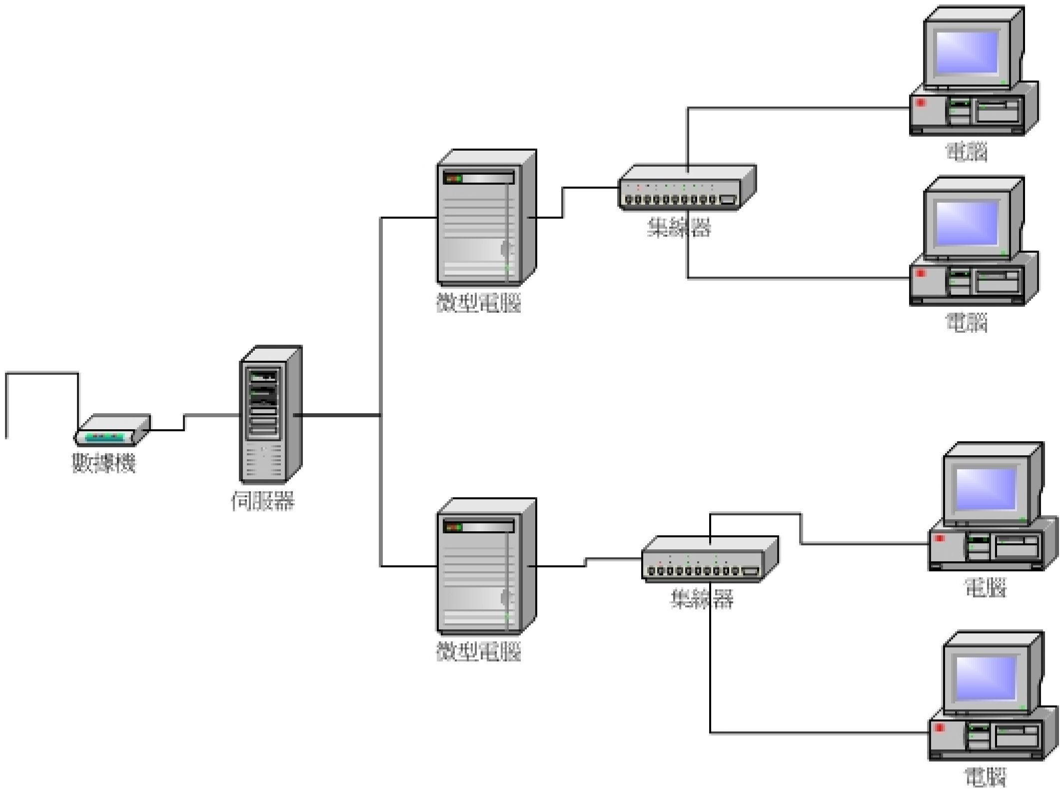
- 既然 ssh 是比较安全的资料封包传送方式，那么我就可以在 Internet 上面开放我的 Linux 主机的 SSH 服务了吗？！请说明您选择的答案的原因！
  最好不要对 Internet 开放您的 SSH 服务，因为 SSH 的加密函式库使用的是 openssl ，一般 Linux distribution 使用的 SSH 则是 openssh ，这两个套件事实上仍有不少的漏洞被发布过，因此，最好不要对 Internet 开放，毕竟 SSH 对于主机的使用权限是很高的！

第十四章、简易 NAT 服务器

- 请简单的说明 NAT 主机的用途与运作原理。
  NAT 主机的最大用途在于『封包伪装』，可以做为内部 Client 主机对外的联机之用(类似 IP 分享器)，当然，也可以做为区域内主机 (DMZ) 的设定！至于其运作原理则主要以 iptables 的过滤机制有关，利用 iptables 来将通过的封包进行 Source IP 或 destination IP 的伪装！

- 假设我是您公司的上层主管，我知道我们单位内共有 100 部计算机 (Windows 操作系统)，其中共分为两大部门，这两大部门的数据是互相独立的，并且两大部门各自拥有一部 Linux 主机在负责档案分享的工作(就是未来会谈到的 SAMBA 主机)。不过，我们公司仅有一条对外的 ADSL 双向 512 的专线而已。现在，我命令您规划全公司 100 部计算机都可以连上 Internet ，而且只多给您一部 Linux 主机，并且原有的 Linux 主机功能 (就是 SAMBA 啦) 还是存在。请问：

1. 公司的网络规划的示意图如下所示：



   两边的 LAN 并没有互相物理连接，此外，微型计算机为本来的 Linux 主机，而新增一部 Linux 主机则是『服务器』那一部！
2. 多出来的那一部 Linux 主机用途为 ADSL 连接与内部私有网络的连接，该 Linux 的网络参数：『IP 为 192.168.0.254(对内)，对外为拨接所产生的接口！启动 NAT 功能』！
3. 两部 Linux SAMBA 主机的网络参数为：『部门 A：IP 为 192.168.1.254(对内卡)以及 192.168.0.1(对外卡)，新增 route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.0.2』，『部门 B：IP 为 192.168.2.254(对内卡)以及 192.168.0.2(对外卡)，新增 route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.0.1』，两部主机的 Gateway 都是 192.168.0.254。
4. 两个部门的网络参数：『部门 A，Network:192.168.1.0/24，Gateway:192.168.1.254』、『部门 B，Network:192.168.2.0/24，Gateway:192.168.2.254』。

第十五章、简易 NFS 服务器设定

- NFS 的主要设定档为何？而在该档案内主要设定项目为何？

  主要的设定档为 /etc/exports 而至于其设定的内容项目在每一行当中则为：
  1. 分享的目录
  2. 针对此分享目录开放的主机或 IP 或网域
  3. 针对这部主机所开放的权限参数！

- 在 NFS 主要的设定档当中仅有少许的参数说明，至于预设的参数说明则没有在该档案当中出现，请问，如果要查阅更详细的分享出来的档案的属性，要看那个档案？
  要查阅 /var/lib/nfs/xtab 这个档案才行！

- 如果已经启动了 nfs 这个服务器，但是却又修改过主要设定档，请问可以使用那个指令来重新挂载分享出来的目录与 client 端权限的设定值？
  透过使用 exportfs ，可以加上 -a 或者是 -rv 这两个参数来重新挂载！如 exportfs -rv

- 在 client 端如果要挂载 NFS 所提供分享的档案，可以使用那个指令？
  那自然就是 mount 啦！还有卸载是 umount 喔！

- 在 NFS 主要设定档当中，可以透过那个参数来控制不让 client 端以 root 的身份使用您所分享出来的目录与档案？
  可以在 /etc/exports 当中的参数项目，设定『 root_squash 』来控制压缩 root 的身份喔！

- 我在 client 端挂载了 NFS Server 的某个目录在我的 /home/data 底下，当我执行其中某个程序时，却发现我的系统被破坏了？您认为可能的原因为何？该如何克服这样的问题，尤其是当我的 Client 端主机其实是多人共享的环境，怕其它的使用者也同样发生类似的问题呢？！
  可能由于您挂载进来的 NFS Server 的 partition 当中具有 SUID 的档案属性，而您不小心使用了该执行档，因此就可能会发生系统被破坏的问题了！
  可以将挂载进来的 NFS 目录的 SUID 功能取消！例如：
  mount -t nfs -o nosuid,ro server:/directory /your/directory

---

第十六章、简易 DHCP 服务器设定

- DHCP 的主要用途为何？
  DHCP 主机的主要用途就是在于自动分配网络参数给 Client 端的计算机，以降低网域当中可能发生的 IP 冲突问题，以及减少网管人员到处检查错误的伤脑筋！

- DHCP 主要的两种 IP 分配模式为何？
  主要的两种分配模式分别为 Dynamic IP 与 Static IP ， Static IP 透过 MAC 的比对，至于 Dynamic IP 则是直接取用网域中尚未被使用到的 IP 来进行 Client 端的分配。

- 在有 DHCP 主机存在的网域当中，且 client 端亦使用 DHCP 来规划客户端的网络参数，那么请问，在该网域当中，Client 端是如何取得 IP 的呢？？
  1. 首先， Client 端会发出一个 DHCP 要求封包；
  2. Server 端接收到要求后，会主动的响应信息给 Client ；
  3. Client 若接受该 DHCP 主机所提供的参数，则主机会记录下租约信息，至于 client 端则开始以主机提供的参数设定其网络；

- DHCP 是如何发送 Static IP 的？可以使用何种指令取得该信息？
  DHCP 主要利用网络卡的硬件地址，亦即俗称的『网络卡卡号』，也就是 MAC 来进行 Client 端的比对的，至于主动取得 Client 端的方式，可以透过 ping 以及 arp 来获得。

- 在 DHCP 的租约档，亦即 /var/lib/dhcp/dhcpd.leases 当中，记录了什么信息？
  这个档案主要记录了 Client 端连上 Server 端的纪录数据，他会被 DHCP 主机用来判定与 Client 端的租约行为喔！

- DHCP 的登录档放置于何处？
  就是最重要的 /var/log/messages 这个档案啦(预设状况下！)

第十七章、简易 DNS 服务器设定

- 为何要有 DNS 系统：
  最主要的功能其实在于 Hostname 对应 IP 的查询，可以让我们人类以计算机主机名称连上 Internet ，而不必背诵 IP 哩！

- 那么请教 Unix Like 系统当中，主要使用那个套件做为 DNS 主机的架设，同时，他又是使用那个 daemon 来启动 DNS 系统？
  在 Unix Like 系统当中，使用 BIND 这个套件做为 DNS 的架设，至于 daemon 则是使用 named 这个 daemon ！

- 最早的 Internet 其实是为了政府人员可以连上网络以进行资源的分享，另外，则是电子邮件的使用。而在早期使用的重要档案只有 /etc/hosts 这个，请教这个 hosts 档案的内容含有什么项目？
  这个档案的『格式』为『 <IP><主机名称><主机别名(aliase)>』，而，这个档案里面放置了至少一行，也就是：
  127.0.0.1 localhost localhost.localdomain
  另外，也可以将经常连接的主机 IP 与 HOSTNAME 的对应给他写进来！

- 请说明 DNS 的三种类型与相关的内容：
  DNS 主机主要分为： master, slave 与 cache-only 三种类型！在 master 当中，master 主机里面即有设定 DNS 数据文件，例如在 /var/named 里面的正反解档案。至于 slave 的 DNS 主机则主要在进行 master 主机的数据备份，同时也提供 Internet 上面的查询功能。使用 master/slave 的最大优点在于「单点维护」的能力！利用修改 master 即可让 slave 的数据同

时更新，减少人力的浪费。至于 cache-only 仅进行快取的纪录，本身并无数据库档案！

- 正解档案(forward)反解档案(reverse)与内部循环使用的档案(loopback)主要的纪录功能为：
  正解文件在设定 hostname 对应到 IP 的纪录，主要的纪录有 A, NS, SOA, MX, CNAME 等等；
  反解文件主要设定 IP 对应到 Hostname 的纪录，主要的纪录为 SOA, NS 与 PTR 等。
  内部循环则是 localhost 与 127.0.0.1 的对应啦！

- 在主要的 DNS 设定档 /etc/named.conf 当中，有一个较为特殊的档案，他的类型为 hint ，请问这个档案的功能为何？
  这个档案主要是由 rs.internic.net 所下载下来的，主要记录了 root (.) 这个 zone 的 IP ！可以让我们的 DNS Server 在找不到数据库时，可以到这个 root 去查询数据！

- 在 client 端搜寻 HOSTNAME 对应到 IP 的查询时,最重要的档案,以及该档案的主要用途为何？
  /etc/nsswitch.conf ：可以用来设定查询主机名称的顺序！例如先查询 /etc/hosts 再查询 DNS 系统；
  /etc/hosts ：最早的名称解析器；
  /etc/resolv.conf：这就是 DNS 系统的 resolver (解析器)了。

- 一般来说，在 Client 端使用的查询 HOSTNAME 的指令大多使用什么？
  nslookup ：可以用来收集一部主机的相关信息；
  dig：可以用来收集详细的主机信息；
  whois ：可以用来收集详尽的 DNS 主机信息。
  host 则较为简单喔！

- 请问 named 重要的信息登录在在那个档案中？
  在 /var/log/messages 当中。

第十八章、简易 WWW 服务器设定

- 请问 LAMP 这个服务器代表什么意思？
  这个名词代表了 Linux + Apache + MySQL + PHP 这个 WWW 服务器的组成！

- Apache 的设定档档名一般为何？
  Apache 的设定档档名为 httpd.conf ，不过，由于 httpd.conf 内容参数可以使用『 include "额外设定档名"』，所以也可能具有其它的额外设定档喔！

- 在 Apache 的设定档当中，哪一个参数是用来设定『主网页』的？
  设定主网页的参数为：DocumentRoot 喔！后面接的是主网页放置的『目录』！

- 哪一个指令用来重新启动与关闭 Apache ？(请以 Tarball 安装的方法来说明)
  其实不论是 RPM 还是 Tarball 都是使用 apachectl 这个档案来启动 apache 的，不过 RPM 已经将该档案整合到 /etc/rc.d/init.d/httpd 里面去而已！

- 当我使用 ps -aux 的时候，发现好多的 httpd... 的程序，这是正常的吗？！最多可以有几个程序是在那个档案的那个参数所设定的？！
  这是正常的，主要由 httpd.conf 里面设定的底下两个参数：
  - MinSpareServers
  - MaxSpareServers

- 又，呈上题，这些程序 (process) 的 owner 与 group 是谁？该察看那个设定档的那个参数？
  同样察看 httpd.conf 里面的 User 与 Group 这两个设定值！

- 如果今天我以 http://your.ip 结果却发现浏览器出现类似 FTP 的画面(会列出该目录下的所有档案)，这是什么原因造成的？该如何避免？
  这是由于在 httpd.conf 里面，针对该目录的设定参数『 Options 』当中，设定了 Indexes 这个设定值，导致当找不到主页时(通常是 index.html)，就会将该目录下的所有档案秀出来！解决的方法就是拿到 Options 里面的 Indexes 设定值即可！

- 在 Apache 里面， .htaccess 这个档案的功能为何？
  可以用来取代 httpd.conf 里面的设定参数！创造属于使用者自己的 Apache 风格！

第十九章、简易 SAMBA 服务器设定

- 一般来说， SAMBA 使用的设定档放置在哪里？档名为何？
  使用的档名为 smb.conf ，通常会放置在 /etc/samba/smb.conf 里面，不过，最好可以使用 rpm -qc packagename 来查询！

- 哪一个指令可以用来判断 smb.conf 这个设定档的正确性？
  当我们修改完 smb.conf 之后，记得要以 testparm 来进行 samba 的确认！

- 哪一个指令可以用来察看 SAMBA 主机分享出什么目录？
  利用 smbclient 即可：『smbclient -L NetBiosName -U username 』！

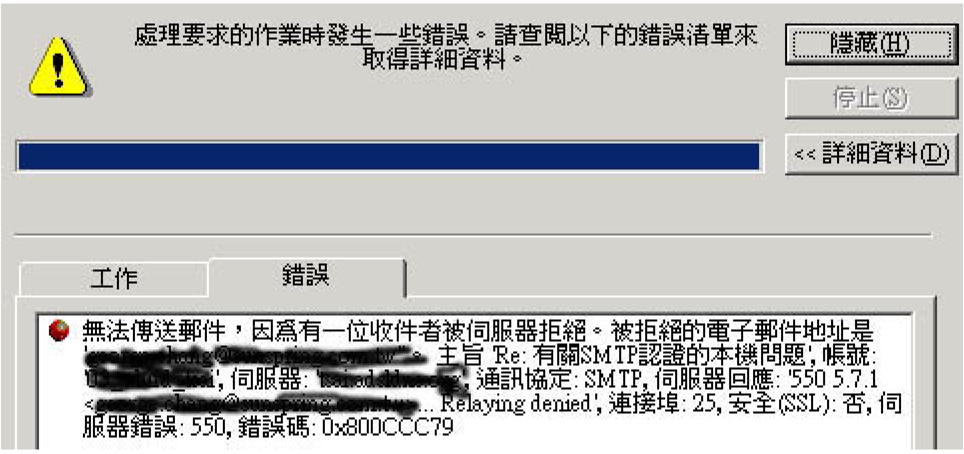
- smbmount 的功能为何？
  在 Linux 系统上面，将 Windows 的网络上的芳邻，或者是 Linux 的 SAMBA 所提供分享的资源

挂载到自己的系统下！

- 我今天使用 smbpasswd 去新增一位使用者 badbird，让他可以登入我的 Linux SAMBA 主机，但是无论如何就是无法新增。您认为原因可能是什么？
  由于 Samba 使用者的信息必须要存在于 /etc/passwd 里面，既然无法新增，应该先确认 badbird 这个使用者已经存在于 Linux 系统当中了！如果还是无法新增，则需要查阅 smb.conf 的设定，看看是否 /etc/samba/smbpasswd 这个密码档案不存在？！

第二十章、简易 Mail Server 架设 -- Sendmail 设定

- 我在使用 Sendmail 寄信时，却发生底下这个问题，请问可能的发生原因为何？



可能的问题有两个：

- 你的 Client 端计算机的 IP 或者是网域地址被 /etc/mail/access 所挡住，或者是 /etc/mail/access 没有打开你的 RELAY 权力；
- 你有进行 SMTP 邮件认证的设定，但是忘记在 MUA 当中设定账号与密码的项目了！

如果不是这两个原因，那么可能的问题就多了！请详细的检查您的网络设定！

- 请列出四个 Mail Server 的相关的组件，以及其功用为何？
  0. Mail Client ：邮件客户端，其实就是使用 mail 的那位使用者所在的计算机即可称为 mail client ；
  1. Mail User Agent ：为一个应用软件，主要的功能就是收受邮件主机的电子邮件，以及提供使用者浏览与编写邮件的功能；
  2. Mail Transfer Agent ：为在计算机与本地端 Mail server 或 Internet 上面的 Mail server 传送讯息与邮件的主机；

3. Mail Dilivery Agent ：主要的功能就是将 MTA 所收受的本机信件，放置到本机账户下的邮件档案中 (Mailbox)！

- POP3 与 SMTP 的功能为何？
    - SMTP 为使用于 MUA 或 MTA 与 MTA 之间的传输协议，通常使用 port 25 ，只要主机支持 SMTP ，并且其它 relay 的条件能配合，就可以进行邮件传递！
    - POP3 可以提供使用者经由 MUA 到 MTA 下载邮件，同时并可将邮件从主机上面删除！

- 请简单的说明 DNS 里面 MX 标志与 Mail 的关系为何？
  MX recode 可以可以让 mail server 经由 MX 以及 A (address) 这个记录来进行 mail gateway 与 mail route 的功能！能够达到的作用相当的多！

- 今天我突然兴起，想要修改我的 sendmail ，请问， sendmail 的设定档在哪里？而我要以什么程序修改 sendmail 呢？
  Sendmail 的设定档为 sendmail.cf ，这个档案通常放置在 /etc/sendmail.cf 或者 /etc/mail/sendmail.cf ！您可以手动的编辑这个档案，不过不建议如此，取得代之的，可以使用 m4 这个程序来进行 macro 的动作，进一步的完成 sendmail.cf 的修改！至于使用 m4 时，需要先建立 m4 scripts ，再以 m4 转换才能形成 sendmail.cf 喔！

- 什么是 mailling list ？在 sendmail 底下有什么方法可以不藉由其它的软件达到 mailling list 的功能？
  Mailling list 就是将使用者寄给一个账号邮件时，该账号会主动的将该邮件传送到所有的用户去!有点类似目前的电子报!在 sendmail 底下,我们可以透过 aliases(需配合 newaliases) 以及 ~/.forward 来达成喔！

- 如何察看邮件队列的内容，以及邮件队列内容放置在何方？
  使用 mailq 即可知道目前邮件队列的内容，而邮件队列虽然可以透过 sendmail.cf 来修改，不过，预设情况下，都是以 /var/spool/mqueue 为邮件队列目录。

- 若我的 sendmail 主机有很多名称，我想让这些名称都可以进行 mail 的接收，应该修改什么档案？
  可以修改 /etc/mail/local-host-names 来达成！

- 什么是 Open Relay？
  所谓的 Open Relay 就是，不论发信端来自何处，您的 Open Relay 的主机均可以帮发信端将信件发送出去，这个称为 Open Relay 。如果您的 mail server 具有 open relay 的情况，那么很容易遭受到垃圾邮件的填充，不但造成网络频宽的耗损，也容易让您的主机被列入黑名单当中！

第二十一章、简易 Mail Server 架设 -- Postfix 设定

- 请问 Cyrus SASL 在 1.5.xx 以及 2.xx 版本中，用来作为 SMTP 的认证的机制有何不同？并请说明不同的版本与 Postfix 的搭配情况。
  一般而言， SASL 1.5 适用于 postfix 1.xx 版本，至于 SASL 2.x 版则适用于 postfix 2.xx 版。由于 SASL 2.x 主要使用 saslauthd 这个 daemon 来做为身份认证的方式，而 SASL 1.5 则使用的是 pam 与 pwcheck 这两个机制，两个版本的认证机制并不相同。所以使用 Postfix 来设定 SMTP 的身份认证时，请务必记得版本的差异！

- 如果要让 Postfix 可以收发来自非本机的外部信件，您可以修改 main.cf 里面的什么参数？
  需要在 main.cf 里面修改的变量主要有：
  0. 当 Client 来自信任的网域，也就是 IP 符合 $mynetworks 的设定值时；
  1. 当 Client 来自信任的机器，也就是主机名称符合 $relay_domains 的设定项目时；
  2. 当 Client 来自不信任的网域，但是去的目的地主机端符合 $relay_domains 的设定时。

- 如何察看您目前的 Postfix 服务器的所有设定参数？(使用什么指令？)
  利用 postconf -n 可以察看『目前 main.cf 里面设定的参数』，而如果要看所有的参数，则直接使用 postconf 即可！

- 在 Postfix 当中，由于已经具有过滤邮件的机制，所以不太需要使用 procmail 了！请问，我该如何启用信件的 Header 过滤机制？同时，如何设定规则，使得 192.168.100.100 这个主机的来信，以及只要邮件的标头为『 Your account 』的信件就予以丢弃？
  启用的方式可以在 main.cf 里面启用这个设定值：
  header_checks = regexp:/完整路径/文件名
  body_checks = regexp:/完整路径/文件名
  至于档案内容可以这样做：(注：因为要查阅的是标头，所以需要 header_checks 后面所接的档名呐！)
  /^Subject.*your\ account/ REJECT

第二十二章、简易 FTP Server 架设 -- Wu FTP 设定

- FTP 在建立联机以及数据传输时，会建立哪些联机？
  需建立两种联机，分别是 command 与 data transfer 的联机，就是命令信道与数据传输信道。一般来说，就是 port 21(ftp) 与 port 20(ftp-data)。

- FTP 主动式与被动式联机有何不同？
  主动式联机的时候， command 联机是由 client 端主动连接到主机端，但是 ftp-data 则是由主机端主动的联机到 client 端。至于被动式联机的时候，则不论 command 还是 ftp-data 的联机，

主机端都是 Listen 客户端的要求的!

- 有哪些动作可以让您的 FTP 主机更为安全 (secure) ?
  有很多的方法，在设定上面有这些方法：
  - 让 guest 与 anonymous 的家目录限制在固定的目录中(chroot 或是 restricted)；
  - 使用 TCP_Wrappers 启动 FTP ，并以他来控制可联机的 IP；
  - 拒绝 guest 与 anonymous 使用 change, delete, rename 等等指令；
  - 拒绝 root 的登入或者其它系统账号的登入；
  - 拒绝大部分的 upload 行为!

- 我们知道 ftp 会启用两个 ports ，请问这两个 port 在哪里规范的？而且，一般正规的 port 是几号？
  是在 /etc/services 里面规范的，此外，正规的 ports 在 command 是 21 而 data 是 20 。

- Wu FTP 的主要设定档在哪里？
  /etc/ftpaccess

- 在 Wu FTP 的设定档当中，那个 log transfer 是干嘛用的？
  log transfer 可以将使用者进行数据传输的时候，将传输的档案大小、内容等等记录到 /var/log/xferlog 里面去!

- 在 Wu FTP 的设定档当中，那个 passive ports 是干嘛用的？
  我们知道 FTP 有主动与被动联机两种方式，如果 FTP 主机允许被动联机，那么 Client 端联机时，主机端将会随机取大于 1024 的 port 来进行 data transfer ，为了避免随机取 port 的问题，所以 passive ports 可以指定固定的几个小范围的 ports ，比较安全。

- 那一个档案可以用来抵挡类似 root 这种系统账号的登入 FTP？
  /etc/ftpusers!(其实是利用 pam 模块来达成的功能!)

- 在 FTP 的 server 与 client 端进行数据传输时，有哪两种模式？为何这两种模式影响数据的传输很重要？
  数据的传输有 ASCII 与 Binary 两种方式，在进行 ascii 传送方式时，被传送的档案将会以文字模式来进行传送的行为，因此，档案的属性会被修改过，可能造成执行档最后却无法执行等的问题！一般来说，ASCII 通常仅用在文本文件与一些原始码档案的传送。

第二十三章、简易 FTP Server 架设 -- Pro FTPD

- 如何建立一个使用者，他可以使用 FTP 的功能，但是无法以 telnet 或 ssh 登入系统？！请使用 proftpd 的设定项目来设定！
  因为无法使用 telnet/SSH ，这表示他可能不具有正常的 shell ，因此您可以将该使用者在 /etc/passwd 里面，最后一个 shell 参数改成 /bin/false ，然后在 proftpd.conf 里面设定『RequireValidShell off』即可！

- 我明明在台湾，我的主机时区 (/etc/sysconfig/clock) 明明在台湾，为何登入 proftpd 之后，显示的系统时间就是慢了 8 小时？请问为什么？如何解决？
  这是由于 proftpd 预设使用 GMT 时间，因此，您必须要重新设定 proftpd ，使得时间使用 localtime 才成！修改 proftpd.conf 里面的参数成为：『TimesGMT off』

- 如果发生了无法登入，或者是与 proftpd 的 FTP 功能相关的错误时，要如何 debug 呢？
  最先应该查询是否被防火墙挡住了(同时观察 Server/Client 端)；
  克服防火墙问题后，再查询 FTP port 是否有启动；
  再检阅 /var/log/messages 里面的错误讯息，
  最后据以修订 proftpd.conf 的设定参数！

第二十五章、简易 Proxy Server 架设

- 请说明为何 Proxy 可以提升网络的 WWW 浏览速度？
  这不但由于 Proxy 可以透过上层 Proxy 达到分流的功能，使得网络传输更具效率之外，还由于 Proxy 会将数据快取(cache)在自己的硬盘内，以方便下次的查询，因此对于局域网络的浏览速度是有帮助的！

- 万一 squid 发生了问题，请问我该如何找出问题点？
  最主要还是在于找出问题，所以需要由 suqid 的登录档查询起，当然，硬盘的优劣以及 squid 所取得 PID 的 owner 与 group 都是需要注意的！

- 请说明 Proxy 服务器的功能为何？
  Proxy 服务器的功能在于代理来自 client 的上网需求，并向目地端主机送出 client 端的要求，以协助 client 端取得所需要的浏览网页。此外，并可将网页数据进行 cache 的功能。

- 试说明为何 Proxy 服务器可以提升网域之内的网络安全性？
  由于 Proxy 服务器通常架设之后，会让网域仅有一个 Proxy 主机的出口，亦即单点对外的服务器，可以强化网络的管理！

第二十六章、简易 Network Information Service, NIS Server 架设

- 请简单说明 NIS server 的功能与工作流程

  当您有多部具有相同账号的 Linux 主机时，即可利用 NIS 所提供的服务，来利用一部 NIS 主机掌控所有的 linux 主机的登入时所需查阅的账号与密码验证。流程如下：

  0. NIS Server 将自己系统内的 /etc/passwd, /etc/group, /etc/hosts 等制作成为 DBM 的数据库格式档案；
  1. NIS Client 若有用户登入的要求时，会前往 NIS Server 搜寻数据库里面的数据做为验证之用。
  2. 每次更动 NIS Server 上面的用户数据时，则 NIS Server 需要重新制作 DBM 数据库档案才行！

- 请简单说明 NIS Server/client 的架构

  NIS master/client 的特色为：

  0. NIS Server 的 master 先将自己的账号、密码相关档案制作成为数据库档案(database file)；
  1. NIS Server 的 master 将自己的数据库档案传送到 slave 上面；
  2. NIS Server 的 slave 接收来自『信任的 NIS Server master 主机』的数据后，更新自己的数据库，使自己的数据库与 master 主机的数据同步；
  3. 网域当中的所有 NIS Client 查寻 NIS Server 时，会找寻『最先响应的那一部 NIS 主机的数据库内容』。

  也就是说，架设 slave NIS server 可以分担区域内 NIS 的工作！

- NIS 启动之前需要先启动那个服务，否则就无法启动成功

  因为 NIS 是 RPC Server 的一种，所以必须要启动 portmap 这个 daemon 才行！

- 我的 NIS 网域名称为 bird ，另外，我主机的 IP 与主机名称为 192.168.5.1/bird.nis.org ，请问要这些信息需要设定在 NIS Server 的哪些档案之内？

  网域名称可以直接手动下达『 nisdomainname bird 』也可以写入 /etc/sysconfig/network 里面『NISDOMAIN=bird』；

  IP 与 主机名称 需要写入在 /etc/hosts 里面。

- /etc/nsswitch.conf 的功能为何？如果我想要让密码查寻先本地的密码文件，再查寻 NIS ，需要如何设定？

  该档案的功能很多，在 DNS 方面，可以用来决定正、反解的顺序，至于密码则可以用来判断何者为先！如果需要先查本机再查 NIS 的密码时，需要的参数：

  ```
  passwd:     files nis nisplus
  shadow:     files nis nisplus
  ```

- NIS Server 将密码等档案做成数据库以提供 NIS client 来查寻，那么请问使用什么动作后，可以将密码档案转成 NIS 的数据库格式档案？

  /usr/lib/yp/ypinit -m

- 如果我想要增加网域当中一个新的账号：newaccount，并且这个 newaccount 可以让 NIS Client 查寻到他的账号与密码，需要进行哪些步骤？
    0. 先登入 NIS Server 以 useradd newaccount 以及 passwd newaccount 来新增账号；
    1. 制作密码数据库：『/usr/lib/yp/ypinit -m』
    2. 重新启动：『/etc/rc.d/init.d/ypserv restart ; /etc/rc.d/init.d/yppasswdd restart』。

- 实作范例题：底下是我的网域参数特征：
  network/netmask:192.168.1.0/255.255.255.0
  NIS server : 192.168.1.100 (hostname: server.nis.test)
  NIS cient: 192.168.1.200 (hostname: client1.nis.test)
  NIS domain name: nis.test
  利用上面的参数来设定 NIS 架构，请一步一步的写下你的设定。
  请自行参考本章节内容设定！

- 承上题：如果我的网域太大了，所以有一部 NIS slave 主机，这部主机的 IP 为 192.168.1.50 ，请问这部主机该如何设定？
  请参考：http://www.linux-nis.org/nis-howto/HOWTO/index.html

第二十七章、简易 NTP 服务器设定

- 什么是 GMT (格林威治) 时间与 UTC 时间？
  由于地球是圆的，所以同一时间点上，在地球共可分为 24 个时区，其中，我们以欧洲的格林威治时间为一个对照的依据，这个即是 GMT 时间。台湾时间比 GMT 时间快了 8 小时。至于 UTC 时间则是由原子钟所计算的时间，这个时间是相当的准确的，主要仍以格林威治时间为时区！

- Linux 系统的所有时区档案放置哪一个目录底下？
  所有的时区档案放置于：/usr/share/zoneinfo 底下！至于系统时区的设定文件则在 /etc/sysconfig/clock 与 /etc/localtime 喔！

- 我的 Linux 主机本来放置在日本东京，现在想将他拿到台湾来运作，不过因为日本与台湾有一个小时的时差，所以我的时间应该需要经过调整才行。不过，因为我的 BIOS Time 主要是依据 UTC 时间来设定的，所以似乎只要更动时区参数即可。请问我该如何设定时区，好让我的 Linux 主机能够显示正确的时间？
  将 /usr/share/zoneinfo/Asia/Taipei 这个档案复制成为 /etc/localtime 即可！

- 目前 Linux 系统上面的时间服务器主要是以 NTP 为主，请问这个 daemon 的主要设定档放在哪里，而该设定档中，针对上层 time server 的设定参数为何？而那个 driftfile 参数是干嘛用的？

在 /etc/ntp.conf 这个档案当中，至于上层 time server 的设定参数为 server 啊！那个 driftfile 则是用来做为『时间差额』的计算的！该参数后面接的是一个完整路径的文件名，该档案里面的数值单位为百万分之一(ppm)。

- 请问 ntptrace 的功能为何？
  可以用来追踪上层 time server 的联机时间与目前时间！

- 我以 date 更新了我 Linux 上面的时间后，该如何将时间数据写入 BIOS 内？
  必须利用 hwclock 这个程序来写入，利用 hwclock -w 写入 BIOS 。

- 在 Linux 上面如何进行网络校时？
  最简单的方法即是使用『 ntpdate time.servers.ip; hwclock -w 』即可！

RootKit Hunter 后端侦测软件之架设与执行

最近更新日期：2004/11/16

由前面几个章节的说明，我们可以晓得因为主机的某些服务是有漏洞的， 黑客们可以针对这些服务的漏洞来撰写恶意攻击的程序，并据以取得该被攻击主机的超级管理员 root 权限。 这些恶意攻击程序后来被散布在因特网上面，因此，很多小朋友很容易就取得这些恶意程序， 并利用这些程序来攻击不特定的众多主机。这种入侵的程序我们可以称之 Root Kit (Root 工具)。 万一您的系统被 root kit 之类的程序所攻击，由于这些程序通常会在您的系统留下一些后门或者是蠕虫， 因此，我们可以透过分析系统来找出这类的程序，这样才能进一步移除恶意程序， 让您的主机保持干干净净啊！底下我们将介绍一套自由软件 RootKit Hunter ， 这套软件可以分析您主机上的可能被恶意程序所攻击的档案，让您可以检查主机是否被入侵喔！

前言：

我们知道，要取得一部主机的所有权限，那就是需要取得该部主机的超级管理员 root 的权限！ 所以一般黑客都会想尽办法去取得 root 的权限的。那么该如何取得 root 的权限呢？ 最简单的方法就是利用网络上流传的 Root Kit 工具程序来进行入侵的动作了。

由于 Root Kit 工具的取得相当的容易，因此难保我们一般使用者的主机不会被低级的怪客所干扰， 所以我们当然要想办法保护我们自己的主机啦！为了要侦测主机是否已经被 Root Kit 之类的程序所攻击， 由自由软件撰写团体所开发的 Root Kit Hunter, rkhunter 这个套件，就能够帮我们侦测啰！ 所以，底下我们就来谈一谈这个咚咚。

什么是 Root Kit

要取得一部主机的控制权，有相当多的方法！最简单的当然就是以登入程序(如 login, ssh, telnet 等等)加上猜测密码的程序来尝试进行登入的行为。不过，由于登入程序大部分都有登入次数的限制， 因此使用密码猜测程序就不这么流行了。

高级的黑客为了系统网络的安全，会撰写一些程序去测试自己主机的服务漏洞， 并且在发现了某些服务的漏洞之后，会通报该服务的维护团体，或者是贡献自己的修补方式， 以补足自己系统的安全性。而服务开发/维护团体在接到这样的通报之后，会在最短的时间内进行程序修改， 并且在因特网上面进行通报与释出该漏洞的修补程序。

然而在这个漏洞通报出来之后，与修补程序释出之前的空窗期，某些恶意的 cracker 就会针对这样的漏洞进行攻击， 这些 cracker 同样是撰写程序来攻击该漏洞，同时取得被攻击主机的控制权，或者是植入木马程序在受攻击的主机上。 这些 cracker 与高级黑客不同的地方，在于他们会很骄傲的将攻击的成果贴在一些 cracker 常上的网站， 藉以推销自己，同时，也会将他们撰写的恶意程序散播到 Internet 上面。有些有心人士就会将这些恶意程序收集起来，做成程序包，并使这些程序包更加流行于 Internet 上面，这些恶意的程序包就被称为 root kit 咯。

RootKit 能作的攻击真是林林总总的说不完！最常见的就是直接以 rootkit 刺探被攻击主机的服务漏洞，如果被攻击主机『刚好』有此漏洞，那么该主机的控制权就可能会被 Cracker 所取得。另外， 若该主机被取得控制权之后，为方便 cracker 未来做为跳板之用，因此他可能会利用其它的 rootkit 将被攻击主机的某些程序换掉，举例来说，我们晓得观察主机的一些信息可以用 ps, ls, top, w 等等的程序，cracker 为了保障自己的入侵不会被真正的系统管理员得知，就会将这些程序换掉， 让原本的系统管理员无法知道目前系统正在跑的程序里面，是否有一些不明的程序存在。

如何防止 rootkit 的攻击

知道了这些 Rootkit 工具包之后，那么我们如何杜绝 cracker 使用 rootkit 程序包来攻击我们的主机呢？ 由于 rootkit 主要是藉由主机的漏洞来攻击的，因此， 您必须要确定『不必要的服务请务必关闭』，此外『随时更新主机上面各套件的修补程序』。 关闭不必要的服务应该很简单，这里鸟哥就不谈了。至于更新套件的修补程序， 最好藉助于 apt 或者 yum 或者您的 Linux distribution 提供的在线更新方式来维护， 这样对于系统管理员来说，会比较轻松。

这样还不够喔！因为 rootkit 也很可能会伪装成 Internet 上面合法的软件， 来吸引您安装他。例如前几年，著名的 OpenSSL 网站上所提供的套件竟然被发现已经被 cracker 置换掉～ 所以，在您安装取得的套件之前，请先以 MD5 或者其它指纹数据进行档案的比对， 以确定该档案是没有问题的。当然，最好还是不要安装来路不明的套件较好。

而为了确认一下我们的主机是否被 rootkit 程序包所攻击， 其实我们还可以透过其它的软件工具来检查主机的某些重要程序，例如前面提到的 ps, top 等等的。 这就是我们这篇文章要提到的 rootkit hunter 啰。

Root Kit Hunter 能作什么？

在官方的数据当中，RKHunter 可以作的事情包括：侦测 rootkit 程序、侦测后门程序、以及主机端的套件检查问题。 rkhunter 所使用的侦测技术包括了底下几种：

- 利用 MD5 指纹分析：
  记得我们在 基础学习篇 里面有提到那个 MD5 的东西吧？简单的来说，每个档案都有自己的指纹数据，这个指纹数据是利用杂凑演算的方式来得到一组 MD5 编码，当这个档案被更动过，那怕是只改了一个字符，而整个档案的容量大小不变， 他的 MD5 编码还是会不同的。因此，若我们在系统安装完毕之后，立即建立重要档案的 MD5 数据库， 然后再以分析工具定期去分析该重要档案的 MD5 编码，若有不同，则显示该档案被变动过， 此时自然就需要了解了解为何会被更动了。

利用这个特性， rkhunter 在释出的时候，就已经收集了各大知名的 Linux distributions 的重要档案的 MD5 编码(例如 login, ls, ps, top, w 等等档案)， 并制作成数据库，然后，当我们安装好了 rkhunter 并且执行之后，他就会利用原本数据库的数据去与我们系统的相关档案进行比对， 若比对的结果有问题，则会显示警示文字，提供系统管理员分析。

- 检查 rootkit 经常攻击的档案：
  如同前面所说的，rootkit 为了伪装自己或者是为了取得系统控制权，他们会主动的去变更某些重要档案。 因此，藉由分析这些档案，我们可以很轻易的就知道该档案有没有被窜改过！ 这也是 rootkit 很重要的一个分析的方法！

- 检查是否具有错误的档案权限--针对 binary files：
  在 基础学习篇 里面的 原始码与 tarball 我们有谈到系统里头真正会执行的其实是经过编译的二进制档案(binary files)，因此， 如果木马程序想要掌握您的系统，那么窜改的那些重要档案自然也就是 binary file 啰， 例如 ls, ps, top 等等的。而重点是，系统原本的这些档案本来都具有比较严谨的档案权限， 例如 /bin/ls 具有的是 -rwxr-xr-x 的 755 权限。不过，很多的木马程序窜改之后的档案权限可能都会变成 -rwxrwxrwx 的 777 权限，因此，直接分析这些重要档案的权限，也可以判断该档案是否有问题。

- 检查隐藏档案：
  有的时候我们为了要让屏幕的显示数据较为干净，可能会将一些档案隐藏起来， 在 Linux 底下的隐藏档案，其实只是在档名最前面加上一个小数点『.』而已。 木马程序也可能透过这个一般朋友们比较不容易注意的隐藏档来隐藏他们的主程序， 因此，rkhunter 也会分析某些不法的隐藏档，以期找出有问题的档案。

- 检查可疑的核心模块(LKM/KLD)：
  在 基础学习篇 里面提到的 核心功能 当中， 可以知道 Linux 的核心功能具有可外挂的特性，也就是 Loadable Kernel Module, LKM 。 而我们也晓得，系统能作什么是由核心来决定的。因此，恶意程序当然有可能藉由加载核心模块来作怪！ 所以啰，rkhunter 也会分析可疑的核心模块。(在 Linux 上面， 我们称核心模块为 LKM,不过,在 BSD 系列的系统上面,他们称为 Dynamic Kernel Linker, KLD。)

- 操作系统的特殊检测：
  每一种操作系统(Operating System)都有他特殊的档案格式,例如 Linux 底下,我们可以使用 ps 来检查 /proc 这个内存目录底下的东西是否一致！？不过，也因为每个操作系统都不相同， 所以这个功能并无法在所有的操作系统上面进行测试的。无论如何， Linux 是有被支援的喔！

- 检查已启动的监听埠号：
  如果要产生网络联机，则在 Server 端需要启动监听的埠号(listening port)， 这样才能监听来自 Client 端的要求啊！这也是所谓的『后门』(backdoor)程序最常用的方法。 我们知道，要启动一个 port 来监听，就必须要执行某个程序才行(基础篇之认识系统服务) 如果我们的系统被木马程序入侵，就很有可能被执行一支程序来启动某项不知名的服务， 而该服务会启动一些 port ，藉由这些 port 就可以让 cracker 轻易的联机到我们的主机。 因此， rkhunter 也会分析主机上面的 LISTENING Ports 来解析是否有问题啊～

- 特定分析(String scanner)：
  某些特定的木马程序或后门程序，他会在系统上面建立一特殊的档案或者是目录， 这些特殊的档

案或目录的文件名是不变的。 所以，rkhunter 会藉由分析这些特定的档案或目录是否在您的系统上面， 以用来判断您的系统是否有被入侵呢？

除了这些方法之外，在新版的 rkhunter 当中，也加入了针对某些常用套件的版本分析。举例来说，Apache 这个套件在 2.0.49 以前的版本已经被发现很多的臭虫， 因此，一般管理者都会建议大家将系统当中的 Apache 升级到 2.0.50 以后的版本(截至 2004/11)。 又例如常见的 SSH/SSL 版本也都有类似的问题。rkhunter 可以分析您系统上面的这些运作当中的套件， 然后告诉您，您的该套件版本是否可能有问题？？但也仅只是『可能』有问题～ 咦！干嘛多了个『可能』啊？呵呵！因为 rkhunter 并不是万能的！底下我们来谈一谈， rkhunter 可能有哪些误判的嫌疑？

---

rkhunter 例外的错误状态

rkhunter 虽然是很棒的一项工具，但是他的输出结果还是有一小部分的问题。举例来说， 在利用 MD5 编码比对方面，因为 rkhunter 是利用他本身的 MD5 编码数据库与您的系统相关档案进行比对， 但是难保您的系统刚好不在 rkhunter 支持的范围之内，如此，则 rkhunter 会判断该档案『有问题！』 此外，如果您是利用 tarball 的方式自行安装类似 syslogd, ps 等档案，由于下达的参数不同， 所以您的这些档案肯定与 rkhunter 的 MD5 数据库不同，此时当然也会被判定是『有问题』的状态。 在这种可预期的情况下，您可以更新 rkhunter 的数据库，也可以与作者联络来克服此一问题。

除此之外，新版的 rkhunter 有提供套件版本的检测，如同上一小节提到的。 但是，各主要 distribution 在发现套件的臭虫后，通常并不是释出最新版的套件， 而是在原有的版本上面透过 patch 来除去该臭虫程序，而并不变更版本。 此时，单纯的检测版本是无法知道该版本有没有经过 patch 的！因此， 如果您的套件版本是已经经过 patch ，但版本却是旧的，此时 rkhunter 的版本检测就会出现错误判断了。

因为如此，所以 rkhunter 在使用上面还是有限制的。如果您想要针对某些服务进行更详细的检测， 那就必须要使用更复杂的程序，例如 nessus 啰！未来我们会再谈到 nessus 的安装与使用方面。

安装 rkhunter：

安装 rkhunter 其实真的很简单！首先，您必须前往下载网页进行下载，下载点：

- rkhunter 下载点： http://www.rootkit.nl/projects/rootkit_hunter.html

在该网页的最下方有个 downloads ，请选择最新版本来下载。鸟哥这里以 1.1.8 版进行说明， 您也可以在 鸟哥的网站上下载 。 假设下载下来的档案放置在 /root 里面，那么整个安装步骤就成为这样(注：您必须要有 bash shell 喔！)

```
[root@test root]# cd /usr/local/src
[root@test src]# tar -zxvf /root/rkhunter-1.1.8.tar.gz
# 此时会产生一个名为 rkhunter 的目录！

[root@test src]# cd rkhunter/
[root@test rkhunter]# ./installer.sh
# 此时会产生一新目录 /usr/local/rkhunter
```

```
# 该目录内含有一些本系统的重要数据，例如 md5 编码的数据等等。
# 另外，检测程序会放置在 /usr/local/bin/rkhunter 喔!
```

这样就安装完毕了!很简单吧!此时我们就可以开始以 /usr/local/bin/rkhunter 这支程序来检测系统了。

检测系统：

系统的检测很简单，因为只要执行 rkhunter 就够了! 与 rkhunter 相关的参数有：

```
[root@test root]# /usr/local/bin/rkhunter --help
# 底下仅列出几个比较常用的参数，更多参数请自行参考!
--checkall (-c)           :全系统检测，rkhunter 的所有检测项目
--createlogfile           :建立登录档，一般预设放在 /var/log/rkhunter.log
--cronjob                 :可以使用 crontab 来执行，不会有颜色显示
--report-warnings-only    :仅列出警告讯息，正常讯息不列出!
--skip-application-check  :忽略套件版本检测(如果您已确定系统的套件已 patch)
--skip-keypress           :忽略按键后继续的举动(程序会持续自动执行)
--quiet                   :仅显示有问题的讯息，比 --report-warnings-only 更少讯息
--versioncheck            :检测试否有新的版本在服务器上
```

那么如何开始检测? 呵呵! 就直接按下 /usr/local/bin/rkhunter --checkall 即可! 例如：

```
[root@test root]# /usr/local/bin/rkhunter --checkall
Rootkit Hunter 1.1.8 is running
Determining OS... Ready

# 第一部份，先进行 binary 的检测，包括 MD5 的检测喔!
Checking binaries
* Selftests
     Strings (command)                                          [ OK ]
* System tools
  Performing 'known good' check...
   /sbin/ifconfig                                               [ OK ]
....(略)....
   /sbin/runlevel                                               [ OK ]
[Press  to continue]      这里按下 Enter 才能继续!
# 在第一部份的检测当中，主要的工作就是在检验一些系统重要的 binary files,
# 这些档案就是常被 root kit 程序包攻击的范围! 所以首先就得要检测他们啊!
# 接下来进行第二部分的检测!

Check rootkits
* Default files and directories
   Rootkit '55808 Trojan - Variant A'...                        [ OK ]
   ADM Worm...                                                  [ OK ]
....(略)....
```

```
  Rootkit 'zaRwT.KiT Rootkit'...                          [ OK ]

* Suspicious files and malware
  Scanning for known rootkit strings                      [ OK ]
....(略)....
  Sniffer logs                                            [ OK ]

[Press  to continue]     这里按下 Enter 才能继续!
# 第二部分就是在检测常见的 rootkit 程序包所造成的系统伤害!
# 这部分的检测当然就是针对各个常见的 rootkit 攻击的档案/目录来侦测啰!
# 接下来是第三部分的检测!

* Trojan specific characteristics
  shv4
    Checking /etc/rc.d/rc.sysinit
      Test 1                                              [ Clean ]
....(略)....
    Checking /etc/xinetd.conf                             [ Clean ]

* Suspicious file properties
  chmod properties
    Checking /bin/ps                                      [ Clean ]
....(略)....
    Checking /bin/login                                   [ Clean ]

* OS dependant tests
  Linux
    Checking loaded kernel modules...                     [ OK ]
    Checking files attributes                             [ OK ]
    Checking LKM module path                              [ OK ]

Networking
* Check: frequently used backdoors
  Port 2001: Scalper Rootkit                              [ OK ]
  Port 60922: zaRwT.KiT                                   [ OK ]

* Interfaces
    Scanning for promiscuous interfaces                   [ OK ]

[Press  to continue]     这里按下 Enter 才能继续!
# 第三部分在检测木马以及可疑的档案属性! 反正就是针对木马程序来进行检测~
# 当然, 因为木马程序可能会开后门, 所以网络服务(port)也在这里检测!
# 同时还包含核心模块等等的检测喔! 再来则是第四部分
```

```
System checks
* Allround tests
   Checking hostname... Found. Hostname is test.vbird.tw
   Checking for passwordless user accounts... OK
   Checking for differences in user accounts...                [ NA ]
   Checking for differences in user groups... Creating file It seems
            this is your first time.
   Checking boot.local/rc.local file...
     - /etc/rc.local                                           [ OK ]
     - /etc/rc.d/rc.local                                      [ OK ]
     - /usr/local/etc/rc.local                                 [ Not found ]
....(略)....
* Filesystem checks
   Checking /dev for suspicious files...                       [ OK ]
   Scanning for hidden files...                                [ OK ]

[Press  to continue]      这里按下 Enter 才能继续!
# 第四部分主要在进行系统开机与相关服务的检测!所以您可以看到
# rc.local 与 password/accounts 的检测都会在这里进行检查~
# 此外,在 /dev 里面也会检查是否有被影响的档案喔!接下来是第五部分

Application advisories
* Application scan
   Checking Apache2 modules ...                                [ Not found ]
   Checking Apache configuration ...                           [ OK ]

* Application version scan
   - GnuPG 1.2.1                                               [ Vulnerable ]
   - Bind DNS [unknown]                                        [ OK ]
   - OpenSSL 0.9.7a                                            [ Vulnerable ]
   - Procmail MTA 3.22                                         [ OK ]
   - OpenSSH 3.7.1p2                                           [ Unknown ]

Security advisories
* Check: Groups and Accounts
   Searching for /etc/passwd...                                [ Found ]
   Checking users with UID '0' (root)...                       [ OK ]

* Check: SSH
   Searching for sshd_config...
   Found /etc/ssh/sshd_config
   Checking for allowed root login...        [ OK (Remote root login disabled) ]
```

```
   Checking for allowed protocols...         [ OK (Only SSH2 allowed) ]

* Check: Events and Logging
   Search for syslog configuration...                        [ OK ]
   Checking for running syslog slave...                      [ OK ]
   Checking for logging to remote system...  [ OK (no remote logging) ]

[Press  to continue]      这里按下 Enter 才能继续!
# 第五部分在检查一些常见的服务的套件版本!
# 因为仅检查版本信息而已，并没有针对可能的漏洞去攻击，
# 所以，这里的信息有可能是 误判的 不要怀疑! 以上面的检测为例，
# 我的 OpenSSL 0.9.7a 是已经经过官方 patch 的版本，也就是说，
# 他已经封住漏洞了，但是这里却显示有问题! 原因就是这样啦!

---------------------------- Scan results ----------------------------


MD5
MD5 compared: 51
Incorrect MD5 checksums: 0

File scan
Scanned files: 328
Possible infected files: 0

Application scan
Vulnerable applications: 2

Scanning took 114 seconds

-----------------------------------------------------------------------


# 最后这里是作一个输出的总结! 我们可以在这里看到
# 最终的简单数据，透过这个数据，可以了解系统目前的状态!
```

在终端机使用 rkhunter 来检测最棒的地方，在于有颜色的显示，以上表来看， 在括号[]内的字样，如果是黄色的 OK 表示没有问题，如果是红色的! 哈哈! 那就表示有点问题了! (在本书上以及网页上的友善打印中，因为打印的问题，所以可能会看不到颜色显示， 很抱歉～没办法～)所以，如果您有看到红色显示的字眼时， 务必特别留意喔!

另外，如果您不想要每个部分都以 Enter 来继续，想要让程序自动持续执行，可以使用:
/usr/local/bin/rkhunter --checkall --skip-keypress
这样就会让程序直接执行到结束喔! 另外，如果想要让程序每日自动执行一次，那就在 /etc/crontab 里面加入这行:
10 3 * * * root /usr/local/bin/rkhunter --checkall --cronjob

以后就会在 3:10 自动执行一次！不过，因为是 crontab 执行的，所以就不会有颜色的显示了。

系统修订：

如果您的系统经过 rkhunter 的检测之后，却发现很多的『红字』时，该怎么办？很简单， 可以参考这个网页提供的方法：

http://www.rootkit.nl/articles/rootkit_hunter_faq.html

基本上，官方网站与一般网管老手的建议都一样，如果被 rootkit 之类的程序包攻击后 (也就是上一节的检测表中的第二部分所攻击时)，那么最好最好直接重新安装系统， 不要存在说可以移除 rootkit 或者木马程序的幻想，因为，『隐藏』本来就是 rootkit 与木马程序的拿手好戏！ 我们不知道到底这个 rootkit 或者木马程序有多剽悍，为了保险起见，还是重灌系统吧！如何重灌？简单的说：

1. 将原主机的网络线拔除；
2. 备份您的数据，最好备份成两部分，一部份是全部的系统内容，越详尽越好，包括 binary files 与 logfile 等等， 至于另一部份则可以考虑仅备份重要的数据文件即可！
3. 将上个步骤的数据备份(仅重要数据部分！)进行整体的检查，察看是否有怪异的数据存在(这部分可能会花去不少时间！)
4. 重新安装一部完整的系统，这包括：
    - 仅安装需要的套件在服务器上面；
    - 先进行 简单的防火墙 设定后才进行联机；
    - 以 APT/YUM 之类的工具进行在线更新；
    - 执行类似 rkhunter/nessus 之类的软件，检验系统是否处在较为安全的状态
5. 将原本的重要数据移动至上个步骤安装好的系统当中，并启动原本服务器上面的各项服务；
6. 以 rkhunter/nessus 之类的软件检验系统是否处在较为安全的环境，并且加强防火墙的机制！
7. 最后，将原本完整备份的数据拿出来进行分析，尤其是 logfile 部分，试图找出 cracker 是藉由那个服务？那个时间点？ 以那个远程 IP 联机进入本机等等的信息，并针对该信息研拟预防的方法，并应用在已经运作的机器上。

这样一来，比较能够保证我们的主机系统可以较为安全一些。至于上头提到的 nessus 软件，我们会在未来几个章节介绍到！

而如果 rkhunter 显示的讯息里面，错误并非是 rootkit 或者木马程序所造成的时候，那么很可能是因为使用者设定上的问题， 或者是系统管理员变动过某些套件所致。举例来说：

- rootkit 显示有怪异的文件名称(strings file)，例如 /dev/.thefile 之类的档案/目录存在，那么首先，您必须要先确定该档案/目录并非是由于 rootkit 所造成的(一般来说，如果 rkhunter 没有在 rootkit 检验部分列出该档案时，几乎就都是这一类的 strings file 啰)， 果真如此，那么就移除该档案吧(确定移除没有问题喔！若不确定，就备份再移除吧～)
- 在 MD5 检验时，发现有 binary file 显示错误！最可能发生此问题的情况，其实不是被入侵，而是系统自动更新套件所致。 鸟哥曾在 Red Hat 9 上面更新过 syslogd 这支程序，没想到 rkhunter 一直显示该档案有问题～ 后来才发现，原来是 syslogd 更新在 rkhunter 之后，而 rkhunter 又没有更新 MD5 编码的数据库， 所以才导致出错的问题。

那如何解决呢？首先，可以透过更新 rkhunter 的数据库来取得最新的信息，如何在线更新？利用：

```
[root@test root]# rkhunter --update
Running updater...

Mirrorfile /usr/local/rkhunter/lib/rkhunter/db/mirrors.dat rotated
Using mirror http://www.rootkit.nl/rkhunter
[DB] Mirror file                     : Update available
  Action: Database updated (current version: 2004081200, new version 2004110700)
[DB] MD5 hashes system binaries      : Update available
  Action: Database updated (current version: 2004091000, new version 2004110900)
[DB] Operating System information    : Update available
  Action: Database updated (current version: 2004091100, new version 2004110901)
[DB] MD5 blacklisted tools/binaries  : Up to date
[DB] Known good program versions     : Update available
  Action: Database updated (current version: 2004091000, new version 2004110500)
[DB] Known bad program versions      : Update available
  Action: Database updated (current version: 2004091000, new version 2004110500)
```

如上所述，我可以将 1.1.8 版本的相关信息 update 到最新的 2004/11/09 所释出的版本！然后再去比对一次 MD5 。 如果这个方法还是无法解决您的问题，就只好请您发信询问 rkhunter 的作者了。

其它的问题解决之道，就请参考上面提供的连结吧！ ^_^y 另外，如果您想要让您的 rkhunter 保持在最新的版本， 利用：
rkhunter --versioncheck
就能够知道目前作者释出的最新版本的 rkhunter 啰！很简单吧！

参考文件

- RootKit 官方网站：http://www.rootkit.nl/

安装 Solaris 10

最近更新日期：2006/08/23

如果您对于在个人计算机 (PC) 的 Linux/Windows 安装很熟悉的话，那么初次接触 Solaris 的安装，肯定会搞的一头雾水， 尤其是在硬盘分割 (partition) 那个地方，这是因为 Solaris 对于 partition 的概念与这两套操作系统并不相同之故！ 另外， Solaris 的安装其实也不能算是很具有亲和力，尤其当无法使用图形接口安装时， 使用纯文字接口的安装可能会是一个很大的挑战啊！ 也就是说，安装 Solaris 还是有一定的困难度！而且， Solaris 原本是仅支持 Sun 自家的硬件而已， 现在虽然已经支持了 x86 的 PC 架构，不过，对于很多在 PC 上面的硬件支持度其实还是有待加强的！无论如何，由于 Solaris 已经支持 x86 的硬件架构了，对于大部分的主流硬件支持度也还算可以， 加上很多大型企业其实使用很多大型主机，所以了解一下 Solaris 提升一下自己的见识与竞争力， 鸟哥觉得，这也是一件不错的事情啊！^_^

安装之前

Solaris 是一套操作系统，这个操作系统可以驱动整个计算机设备，让使用者可以操作计算机设备来达成他们的工作。那么计算机设备里面有哪些组件？操作系统如何与计算机设备搭配？这些我们都得要先了解一下， 然后再来谈开始安装 Solaris 的啦！那学习 Solaris 有啥好处？你该不该学会 Solaris 呢？ 赶紧来去瞧一瞧！

什么是 Solaris

Solaris 是一套操作系统，但是这套操作系统是怎么开发出来的？ 他适合在什么机器上面运作？他与 Unix 这个操作系统又有何相关？都值得来讨论讨论！

- 关于计算机硬件

有人说，计算机是很厉害的咚咚，鸟哥认为，计算机只是一个很厉害的工具，其实他很笨的！ 因为如果你没有下达命令给计算机作，他就不会作任何事情啊！ ^_^ ！ 目前的计算机只认识 0 和 1 ，利用 0/1 的方式来帮助人类进行运算或者是处理其它的事务性工作。那么计算机是什么呢？就如同你看到的个人计算机一般，计算机主要分为：输入单元(键盘、鼠标)、 中央处理器(主机机壳内的 CPU、内存等)、输出单元(例如屏幕、打印机)等等， 当然还有储存设备例如硬盘、软盘等等组件。

时至今日你会觉得『啊计算机不就是个人计算机，还有什么不一样吗？』，当然不一样～ 现今的计算机硬件主要分为 x86 架构的个人计算机以及 RISC 架构的大型主机 (mainframe)。 早期个人计算机尚未流行的时候，主要的计算机架构是 RISC 这种架构的。

在 RISC 的架构中，由于开发商的不同，所以硬件相关规格或多或少就有点不兼容。 举例来说， Sun 的主机与 HP 的主机彼此之间就无法互相交换使用。 RISC 架构的主机早期是很流行的，这是因为该架构可以具有多任务处理的能力，让该架构可以负责负载较重的任务。 不过近来由于 x86 架构的高速成长，让个人计算机的多任务处理能力并不会比 RISC 架构逊色。

由于硬件开发商在发展 RISC 架构的硬件通常是全部的组件整合开发的， 所以各组件之间的整合性理论上会比较好。而个人计算机的 x86 架构是较为开放的， 只要符合 x86 架构的硬件配备，理论上就能够搭配使用。例如你可以使用 NVidia 这家公司发展的主机板芯片组， 使用华硕制造的主机板，配合 AMD 制造的 CPU 还有创见的内存，搭配 WD 的硬盘等等， 最终组成一部 x86 个人计算机。不过，由于制造商众多，在整合上面或许有些小问题也说不定。

就因为大型企业最重要的任务是『稳定的提供服务』，所以 RISC 架构的大型主机还是有其存在的必要！ 不管怎么说，各种架构都存在而各有其适用性，也是一件不错的事情啊！ 那么接下来我们要来谈一谈，『如何使用你的计算机』？ 咦！啊不就按下电源就能够玩计算机了？

- 关于操作系统

当你按下计算机主机的电源之后，计算机会开始读取硬盘内的数据，然后『驱动所有的计算机硬件配备』，包括 CPU、内存、硬盘、网络卡、周边接口等等。之后再加载一些应用程序， 接下来你就可以使用这些应用程序来处理你日常的工作了！ 这些『驱动硬件的程序、应用程序』等等，可不包含在硬件内的啊！ 这些是所谓的『软件功能』，而『 驱动所有的硬件配备，就是操作系统最底层的核心的重要功能』了！ 这样您就可以了解啥是操作系统了！ ^_^

如同前面提到的，早期由于各家硬件厂商开发的硬件规格上或多或少都有点不相同， 那么『驱动硬件的程序自然也就不一样』了！ 所以说，各家硬件厂商也必须要自行开发可以驱动他们家硬件的操作系统才行。 这也就是说：『操作系统与硬件是有相关性的』啊！ 但如此一来也就造成很多的困扰，怎么说呢？看看底下的例子：

由于应用软件必须要使用到操作系统所提供的相关功能，所以在不同的操作系统上面是没有办法执行相同的一套软件的。 也就是说，如果我是软件开发商的话，那我想要让我的软件可以在各家主机硬件上面跑的话， 我就得要重写我的软件，让他可以在不同硬件平台上的操作系统里面执行， 哇！那光是写不同版本的程序，就可以让人疯掉了就要发疯了呐！

这个情况也就造就了在大型主机上面有很多『专属软件』的现象，因为这些软件只能在某一些操作系统上面执行的原因！ 这样说，您对于硬件、操作系统与软件应该有一定程度的了解了吧？

- Unix 、 BSD 、 SunOS 及 Solaris

各家硬件厂商自行以某些独特的方法开发自家的操作系统，一直到 1973 年以后才比较好一点！因为 1973 年以 C 程序语言写出来的 Unix 操作系统被释出了！由于 C 程序语言与硬件并没有直接的关系， 只要你有程序语言编译器，那么透过修改原始码就能够重新编译出适合您硬件的操作系统了！ 在此要再强调一次，计算机硬件仅认识 0/1 这种二进制的数据，而人类对于二进制并没有很深的概念， 所以人们便透过：

- 利用程序语言的语法撰写程序代码，这个程序代码通常是纯文字的数据，所以人们可以很轻易的看懂；
- 这个原始码(就是程序代码)需要经过编译 (compile) 后才能够成为计算机能够认识的二进制 (binary) 档案。

而 Unix 既然有释出原始码，那么只要你将原始码透过一些修订以符合你硬件所需要的规格， 呵呵～那就可以编译出适合你机器的操作系统了！而且 Unix 的速度、概念、效能都很好， 也就造成大流行。当然啦，很多硬件开发商也就直接利用 Unix 来进行操作系统的开发！ 对于软件开发商来说，既然都是使用 Unix ，并且大家都遵守一些工业规范的话， 那么软件的开发也就变的更简单了！

到了 1977 年，Unix 传到加州柏克莱 (Berkeley) 大学，被 Bill Joy 改版成为 BSD (Berkeley Software Distribution, BSD) 的版本，利用 Unix 的概念但是舍弃了原本 Unix 的程序代码， 因此 BSD 虽然是 Unix 的一个分支 (Unix-like)，不过却拥有自己的版权喔！ 后来 Bill Joy 自行创组升阳公司 (Sun) ，并且将 BSD 改成 SunOS 这个操作系统， 并且在 SunOS 5.x 版后重新将他们的操作系统更名为 Solaris！

早期的 SunOS 或者是 Solaris 仅能在 Sun 自己开发的硬件上面跑而已， 他们并没有释出其操作系统的原始码。但是 Sun 的机器在大型企业以及学术单位使用的相当广泛， 包括鸟哥之前所待的研究室里面就有一部 Sun 的主机，而且很多学术界、工程界所开发的软件都是仅能在 SunOS 上面跑而已，所以，Sun 的机器与 Solaris 操作系统对于某些人来说是相当重要的！

近年来由于自由软件的风行， Solaris 也搭上这趟顺风车，此外， Sun 也将 Solaris 移植到了 x86 的个人计算机架构上面了！也就是说，您现在可以使用个人计算机跑 Solaris 啰！ 这真是个好消息！因为很多的大型企业或者是学术单位等等，都还是有使用 Sun 相关的机器， 但是 Sun 的 Solaris 与目前最流行的 Linux 是有些不太一样的地方，包括各项装置代号、开机流程、 对于硬盘分割的概念不同等等，所以还是得要花时间去熟悉他的。早期因为 Solaris 仅能在 Sun 的机器上跑，所以我们也没有办法学习(因为 Sun 的机器很贵！)，现在既然可以在个人计算机上跑， 呵呵！当然得要给他学习学习！除了增广见闻的主要目的之外，多学一样工具，总是对自己的竞争力多一份加分啊！ 您说是吧！ ^_^

另外， Sun 除了将 Solaris 释出成为 Open Source 的软件之外，较知名的其实是释出 StartOffice 成为 OpenOffice 这个软件，对于目前的 Linux 桌上用办公计算机其实是相当有帮助的呐！ ^_^

所以说， Solaris 就是一套操作系统，目前这套操作系统已经支持 x86 了， 这套系统上面还有很多 Sun 自己开发的软件以及工具，大致上就是如此啦！

所需最小硬件的需求

想要安装 Solaris 在你的个人计算机上面时，得先来了解一下你需要什么等级的计算机配备才行。 由于 Solaris 实在.....有点慢，所以你的硬件实在不能太差！简单的说，你应该要：

- 最好能够有 CPU 为 P-III 等级以上的主机架构；
- 至少需要 256 MB 的物理内存容量；
- 最好能有 5-7 GBytes 以上的硬盘空间；
- 若要进行多重操作系统的安装时， Solaris 仅能安装在所谓的『Primary partition』当中；
- 最好能够有较为平常的硬件装置，不要使用特殊的硬件装置。

上面的配备在 2006 年的现在你或许觉得那又没什么，新买的计算机都要比上头列出的配备要更佳！不过，如果你跟鸟哥一样都是喜欢捡旧的货色来做成服务器的话，那么就得要先将你的主机配备找出来看看啰～ 如果没有上述的配备，那最好不要安装 Solaris 了～因为.....真的有点小慢！

另外，如果你想要使用你的个人计算机做成多重操作系统的话，那么那个『 磁盘分割 (Partition) 』要很注意！ 因为 Solaris 的磁盘分割与传统的 Windows/Linux 并不相同，这点在后续我们也会继续的来介绍的喔！

硬盘的磁盘分割

现在的操作系统由于包山包海，除了主要的驱动程序之外，其它的应用软件也都加入很多很多， 所以传统的磁盘已经无法适用于开机启动操作系统了！因此就得要藉由所谓的硬盘来储存操作系统所需要的数据。 在 Unix Like 的操作系统当中，任何装置都是以档案的型态来代表的， 那么在 Solaris 当中硬盘的代号又是什么呢？我们知道现在主机常见的硬盘接口主要有 IDE, SCSI 以及 SATA，在 x86 个人计算机的环境当中则以 IDE 以及 SATA 接口为主的。 这些装置在 Solaris 的代号并不相同。

在 SATA 尚未成为主流之前，IDE 是最主要的硬盘的数据传输接口。实时到目前的环境当中， 为了让主机能够支持传统的 IDE 装置 (硬盘、光盘、刻录机等) ，所以主机上面还是会有 IDE 的连接接口。 咦！没见过 IDE 吗？呵呵！那玩意儿就是很宽的排线所连接的设备啊！通常主机上面都会有两个 IDE 插槽， 每个插槽所连接出来的排线可以连接两个 IDE 装置，这两个装置称为 Master 与 Slave ， 而他们在 Solaris 当中的代号是这样的：

IDE 插槽 排线	Master	Slave
IDE 1	c0d0s[0-15]	c0d1s[0-15]
IDE 2	c0d2s[0-15]	c0d3s[0-15]

很奇怪的代号吧！^_^！其实也没有这么难记啦！你就这样记得好了：

- c ：意义就是控制器 (controller) 的缩写啦！ c0 就是第一个控制器；
- d ：意义就是磁盘 (disk) ， d0 就是第一个磁盘；
- s ：意义就是分割区 (slice)， s0 就是第一个分割区，最多由 0~15 号！

所以说，主机上面的第一个 IDE 控制器（其实通常只有一个啦！）的 IDE1 所接的 master 磁盘， 里头的第一块 Solaris 分割区就被设定为： c0d0s0 啰～这样就不会太难记了吧？ 那如果是 SATA 的硬盘（与 SCSI 相同！）的话呢？通常主机上面应该也是只有一个 SATA 的控制芯片， 而这个控制芯片在主机上面可以提供多个 SATA 硬盘连接的插槽，每个插槽都有其代号， 所以他的硬盘代号则为：

SATA 插槽	sata1	sata2	sata3	...
装置代号	c0t0d0s[0-15]	c0t1d0s[0-15]	c0t2d0s[0-15]	...

同样的，也没有那么难记！你依旧可以这样记忆：

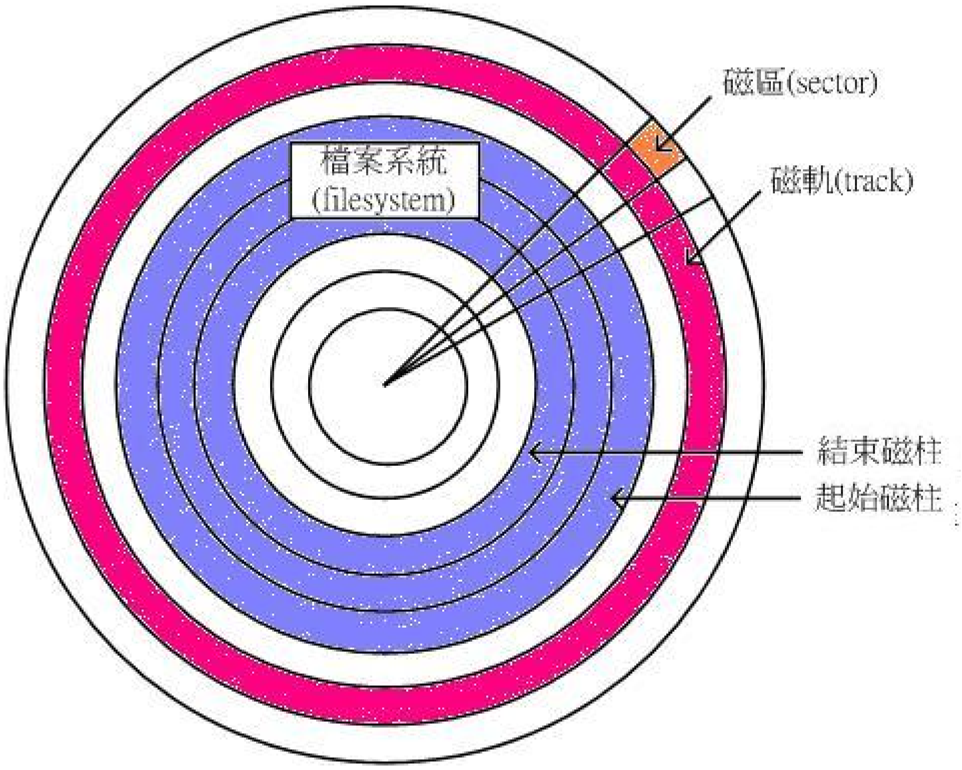
- c ：控制器 (controller)；
- t ：插槽代号 (target number) ，例如 SATA 第一个插槽 (stat1) 就是 t0 ；
- d ：在该插槽上的第几个磁盘，第一个磁盘就是 d0 ；
- s ：同样的啊，分割区嘛！

所以啰，第一个 SATA 控制器所提供的第一个 SATA 插槽，上面接的硬盘的第一个 solaris 分割区就是 c0t0d0s0 的啦！第二个 SATA 的插槽代号则为： c0t1d0s0 喔！ 不同点在于 target number 的啦！

谈完硬盘的代号后，接下来谈一谈： 那么硬盘里头有什么呢？其实如果将硬盘拆开的话，您会发现里面最重要的就是几个组件：

- 磁头 (head)：用来读取与写入磁盘盘上头的数据；
- 扇区 (sector)：硬盘最小的储存单位，每个扇区为 512 bytes；
- 磁柱 (Cylinder)：顺着同心圆转一圈的立体空间，这也是磁盘分割槽的最小单位。

整个硬盘最重要的就是那个纪录数据的磁盘盘，磁盘盘就有点像底下的图示：



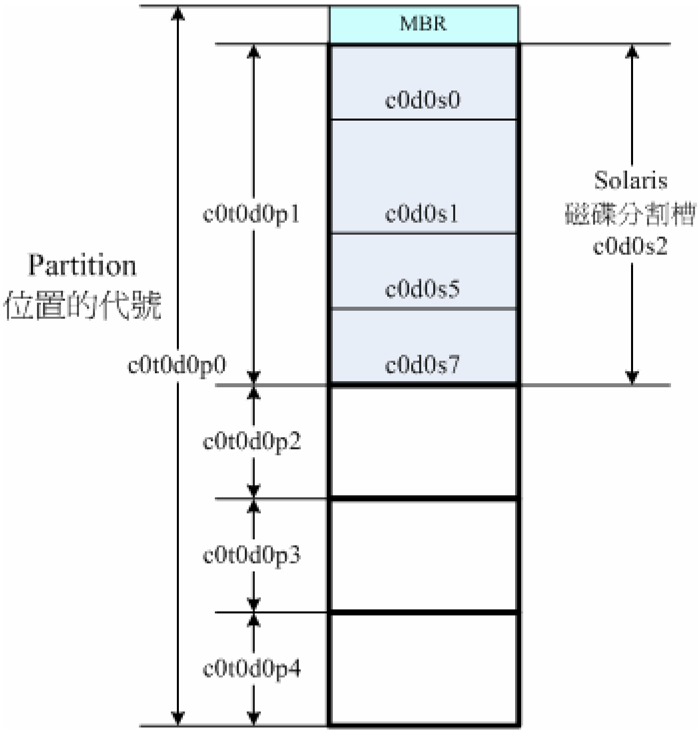
图一、磁盘盘的示意图

除了磁头、扇区、磁柱之外，在整个磁盘盘的第一个磁柱的第一个扇区，也就是被称为主要开机扇区 (Master Boot Record, MBR) 记录了这个磁盘的最重要参数，也就是磁盘分割槽的定义。 而每次在使用该磁盘时，这个所谓的 MBR 一定会被读取，所以这个 MBR 如果坏掉了， 那这部硬盘机也可以宣告寿终正寝了！

MBR 主要记录了两个最重要的信息，一个是开机管理程序 (boot loader) ，这就是当我们在开机的时候，屏幕会出现 Windows 或 SPFdisk 或 Linux 的 Grub 等选单程式的安装处！ 另一个则是这部硬盘的磁盘分割纪录。 什么是磁盘分割呢？如果你用过 Windows 的话应该会知道，明明只有一颗硬盘，不过我们就是可以拥有 C:，D:，E:.. 等等的磁盘槽，那个磁盘槽的定义就是写在 MBR 里面的啦！ 而由于 MBR 是一个扇区的大小 (sector 一个为 512 bytes) ，里面最多仅能记录四笔磁盘分割槽的纪录， 这四个记录是硬盘本身的物理定义，不能改变的。这就是整个硬盘最主要的概念呢！

磁盘分割 (partition) 是个很重要的任务。但如前面提到的，硬盘的物理限制上最多就只有四个分割信息，那如果你的硬盘想要分成四个以上的分割槽时，那该怎么办？这个问题在 Linux 与 Solaris 之间的处理方式不太相同。 Linux 这个操作系统使用类似 Windows 的概念，将那四个 partition 其中一个定义为延伸分割， 而延伸分割的数据则指向其它的扇区额外的定义其它的分割槽，因此延伸分割需要再处理成为逻辑分割后， 才能够被 Linux 操作系统所利用。

但 Solaris 则不是这样子定义，Solaris 仅会使用到 MBR 里面的四个磁盘分割记录的其中一个，而该记录其实是指向 solaris 的磁盘分割定义区， 在 solaris 的扇区内需要再被分割成为更多的磁盘分割槽！ 看不懂吗？让我们用底下这张图来介绍一下：



图二： Solaris 的磁盘分割概念

也就是说，虽然 MBR 可以拥有固定的四个 partition 记录，不过，Solaris 仅能利用其中的一个记录，其它的三个可以保留给其它操作系统使用。这个很重要！ 如果你的 x86 个人计算机当中想要安装多重操作系统的话，而你的剩余空间是放置在 Linux/Windows partition 的 Logical partition (逻辑分割区)，那很抱歉， Solaris 是无法安装的！

上面图一的地方，如果你看不清楚的话，那简单的说，你可以这样看：

- 硬盘的架构内，由于 MBR 仅有 512 bytes 的关系，所以最多仅能有四笔 partition 的纪录， 所以你当然可以看到图一左边的介绍当中，就只有四个主要的纪录区块了！ 再次强调， partition 的最小单位是磁柱 (Cylinder) 喔！

- Solaris 在 x86 上最多仅能使用到一个区块，在图一的例子当中，我们的 Solaris 使用掉第一个区块；

- 在该区块当中， Solaris 可以继续的进行磁盘分割，分割出来的区块就是 Solaris 可以使用的磁盘分割槽了。 在这样的磁盘分割槽当中，以 IDE 硬盘为例，最多仅能分割出 16 个区块，亦即是 c0d0s0, c0d0s1..., c0d0s15； 但实际上，可以使用的分割仅有 c0d0s[0,1,3,4,5,6,7]，这部份得要清楚清楚喔！也就是说，Solaris 目前并不提供大于 s7 以上的分割区 (slice) 啦！

- 在所有的分割槽当中需要注意的是那个 c0d0s2 了，那个磁盘分割槽不可被移除或修改， 因为那个就是来自 MBR 的记录，被称为 overlap ，你当然不能变更他了！

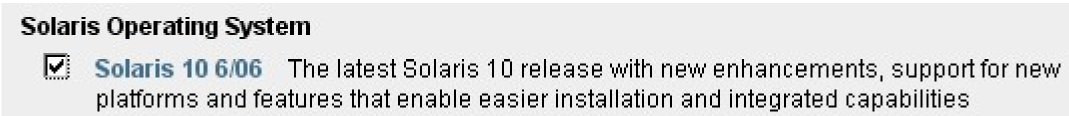
这样说明的话，比较容易理解了吧？上面的分割也看懂了吧？这个真的很重要！ 如果不了解的话，后面提到的实际安装流程就会看不懂啦！切记切记！！ ^_^

下载 Sun Solaris

既然要玩 Solaris ，首要工作当然就是下载啰！ 目前 Solaris 提供 CD, DVD 的格式给使用者，鸟哥的主机因为没有 DVD ，所以选择下载 CD ， 不过，Solaris 10 总共需要 6 张 CD ....实在是有点太多了吧～ 无论如何，就给他下载了先！你可以到底下的网址进行下载：

- http://www.sun.com/software/solaris/get.jsp

选择底下的图样：



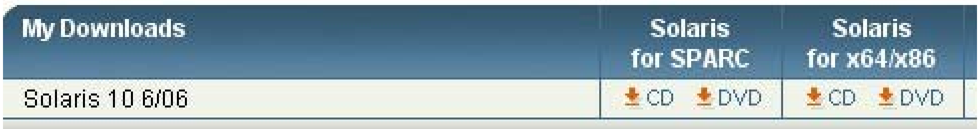
图三：选择下载的数据

然后在该连结往下方查阅，会看到如下的按钮，按下『Get Downloads & Media』：



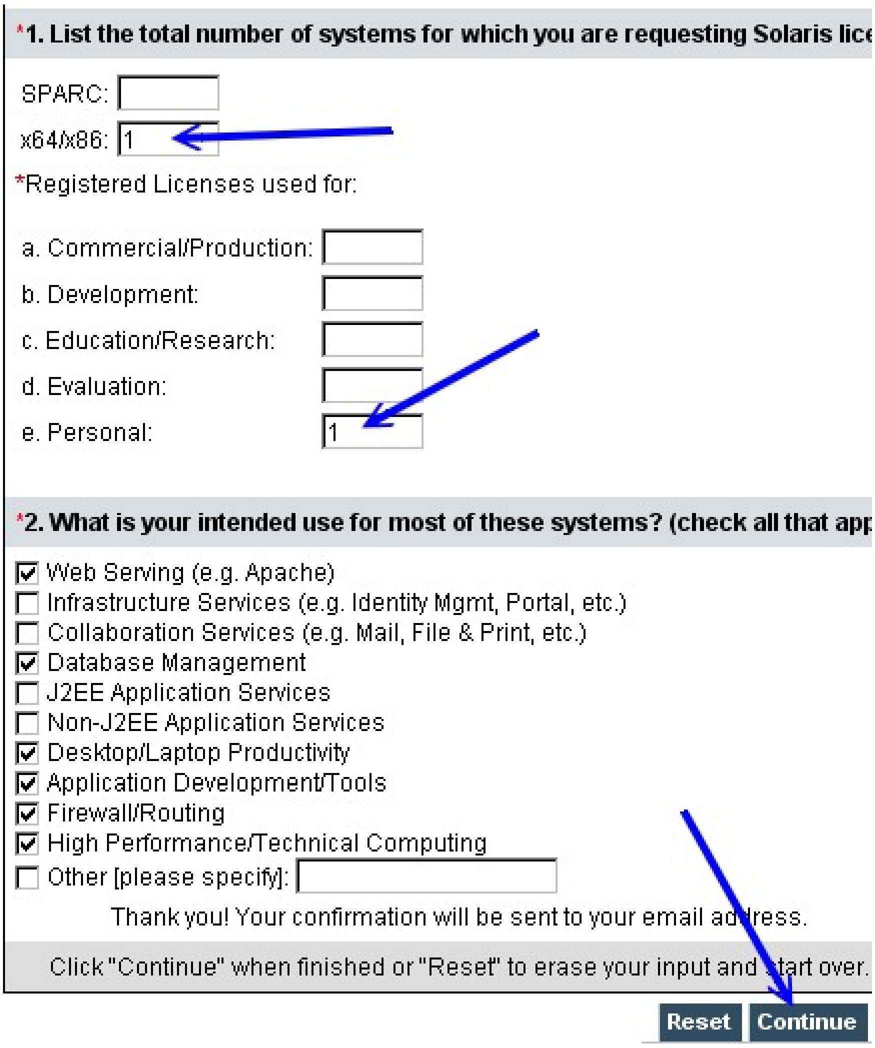
图四：选择下载的数据

在经过了注册 (register)、登入 (login) 后，就可以让你开始准备下载了！ 不过，你必须要在出现的网页当中接受一个授权 (License) 后，才可以开始下载。 刚刚说过，由于鸟哥是使用 x86 的主机架构，同时仅有光驱，所以如下图：



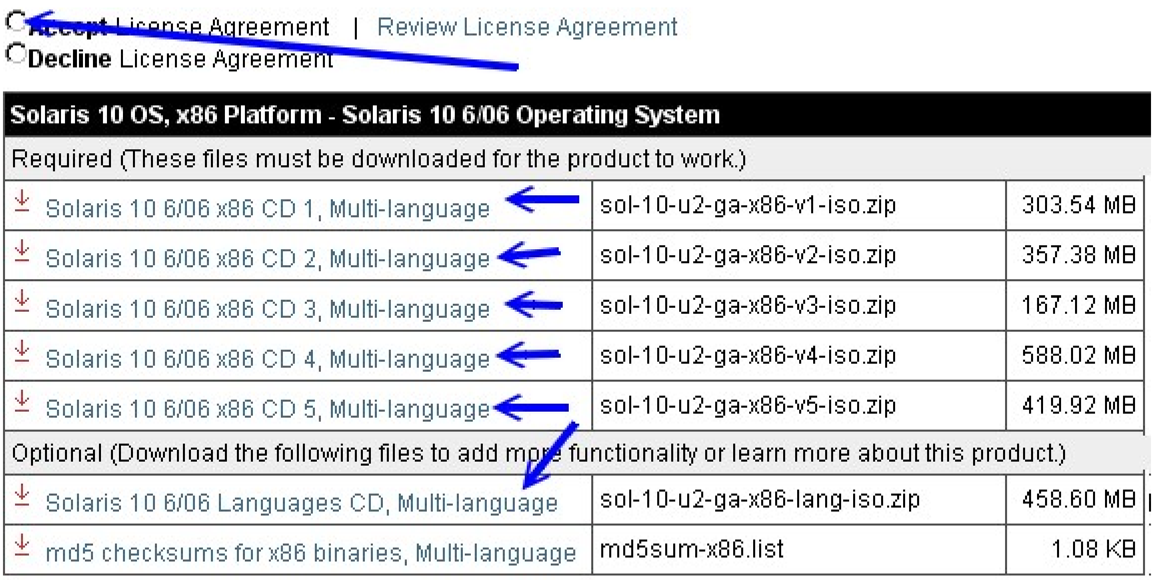
图五：选择所需要的下载数据

鸟哥当然选择上图最右边的 CD 项目来下载了。如果你的环境或者是公司使用的是 Sun 的主机硬设备，由于 Sun 的硬设备通称为 SPARC ，因此，你当然就需要下载 Solaris for SPARC 啰！ 反正，就是依据你的主机来下载相对应的数据啦！之后出现的画面如下：



图六：选择所需要的下载数据

依据你的意愿来填写相关的数据，最后按下『 Continue 』就能够看到一堆连结：



图七：选择所需要的下载数据

不过记得在开始下载前，请先按下上图第一行的『 Accpet 』那个小圆点，否则不能开始下载喔！ ^_^！ 而且也要记得总共有 6 片光盘需要下载的！下载完毕后给他开始烧录起来，那就可以开始安装了！

开始安装 Solaris

好了，讲了这么多的基本数据后，再来则是要开始实际的来安装你的 Solaris 啦！如前面提到的，Solaris 毕竟原本仅是针对 Sun 自家的 sparc 硬件所规划的操作系统， 虽然已经转成可以支持 x86 ，不过总是在支持度上面会有一些落差。因此，您最好不要使用冷门的设备比较妥当啊！ ^_^！ 所以，底下鸟哥会先介绍一下自己测试用的硬设备，再来谈详细的安装流程。

鸟哥的主机配备

在底下的安装范例当中，鸟哥这部主机的定位是『练习用主机』，所以里头预计仅有 Solaris 系统， 并没有要做成多重开机喔！鸟哥使用的是 Asus 的准系统，内部使用 celeron 1.2 GHz 的 CPU， 共有 512 MBytes 的内存容量，另外使用的是内建的 Sis 显示卡， 因为是内建的显示卡，所以与主存储器共享一些内存容量，鸟哥利用 64MB 作为显示卡内存， 所以主存储器仅有 (512-64=448MB) 左右而已。另外这张主机板有内建网络卡。不过内建的网络卡 Solaris 捉不到，所以外接卡部分仅多了一张 Realtek 的网络卡，这张卡就是有名的『螃蟹卡』啦！

在储存设备方面，这部准系统含有一部 52 倍数光驱，一部软盘机，以及一部抽取式硬盘盒，鸟哥使用 40GB 的硬盘来使用这部准系统。预计将这部硬盘拿 20GB 出来安装 Solaris，而其它的 20GB 则做为其它的练习之用。

老实说，这样的硬件配备对于 Linux 来讲， 已经好到不得了了～不过对于 Solaris 来讲，这样的配备是差不多可以顺畅的使用而已，如果要更顺畅的话，最好是使用时下入门级以上的个人计算机来安装比较好，例如 AMD 的 K8 系列，就是一个不错的解决方案啊！ 不过，越新的配备可能又会让 Solaris 捉不到 (这个问题在每个操作系统都可能会遇到！)， 真是陷入一个两难的境界啊～～ @_@

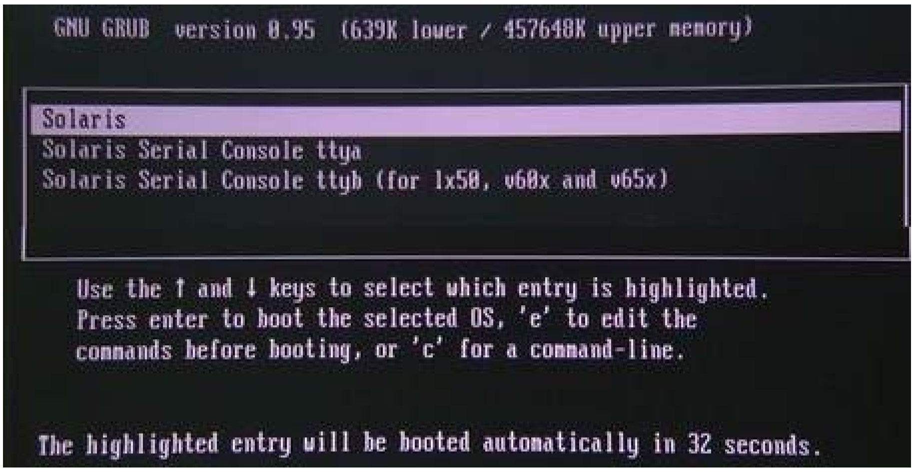
准备开始安装

安装的过程当中，最重要的就是『硬盘分割 (partition)』、『安装开机管理程序 (boot loader)』以及 『软件的挑选』等等，在 Solaris 的安装过程当中，同样的也是这三个重点最重要！ 但是在开始这三个项目之前，我们得就整个系统的环境来作个定义，包括语系的选择、屏幕分辨率的调整等等。 在这个步骤咱们就来看看这些简易的定义吧！

---

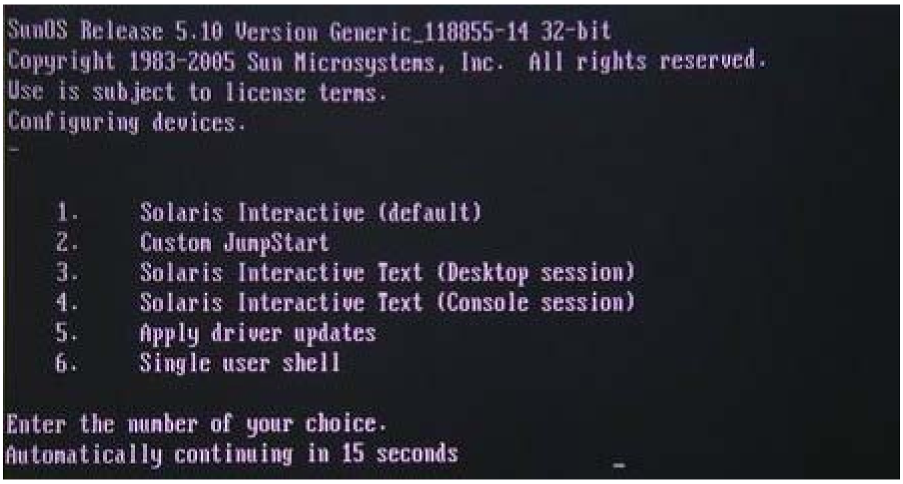
1. 利用光盘开机进入安装画面：

   你必须要调整你的 BIOS ，就是在开机的时候按下『 Del 』按键进入一个可调整主机整体设定值的地方， 并指定让光驱开机，然后放入刚刚烧录起来的 Solaris 光盘，重新开机之后就能够出现如下的图示：



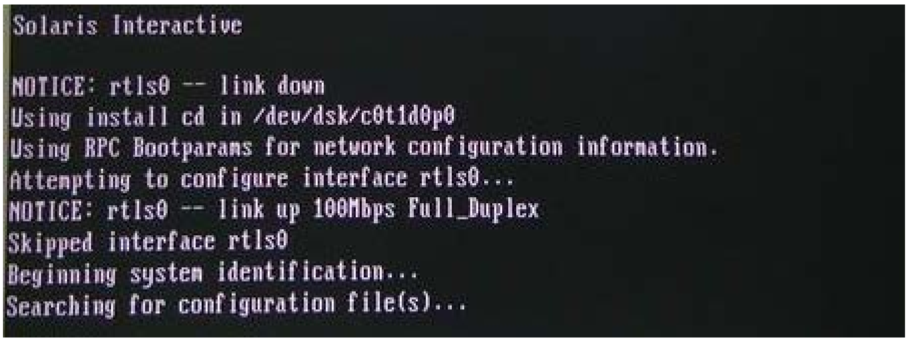
图八、准备开始安装

预设这个画面会等待 60 秒，如果不想要等待的话，直接按下第一个『 Solaris 』就能够进入下一个挑选安装方式的画面了！



图九、准备开始安装

上面的图示中有许多的可用安装方式，你可以输入 1-6 ，建议直接使用『 1 』这个基本的安装画面来安装即可。 这个『 1 』的选项预设是可以使用图形画面来安装的！不过，如果 Solaris 捉不到你的显示卡， 依旧会使用纯文字的方式来进行安装啊！当你选择了安装方式后，屏幕会闪过类似底下的画面：

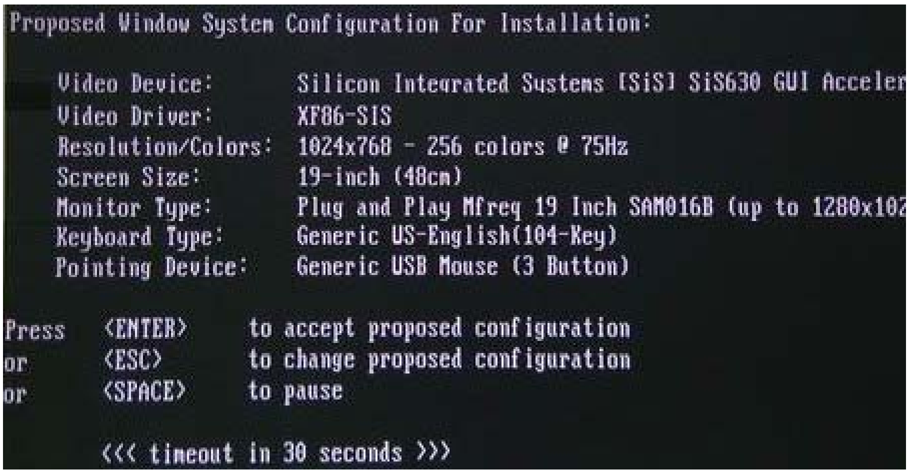


图十、准备开始安装

---

2. 屏幕显示的选择：

上一个图标显示的过程当中，主要在侦测你的一些基本的主机硬件，所以会花掉一些时间，在鸟哥安装的过程当中， 上面这个图示似乎花掉了鸟哥大约 5 分钟的时间喔！等到硬件侦测完毕后，如果顺利的话应该就可以进入图形画面， 不过.....鸟哥的主机环境似乎不很好，结果 Solaris 捉不到这部主机的显示卡， 所以系统会出现一个确认的画面跑出来，如下所示：



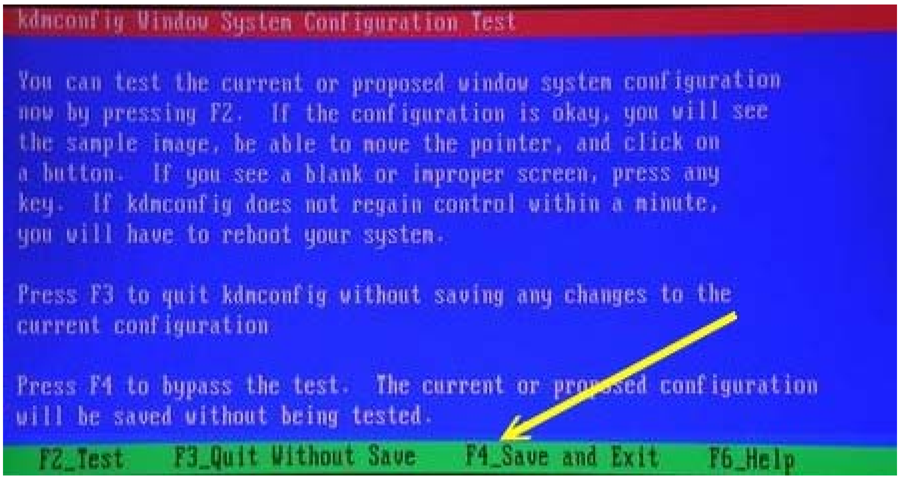
图十一、准备开始安装

在上面的图示当中，你可以直接输入『Enter』就可以进入屏幕相关选项的修改画面， 如下所示：



图十二、准备开始安装

鸟哥的这个系统颇奇怪，看看上面的图标显示的数据都与鸟哥所使用的硬件相符合，不过， 就是无法顺利的启用图形接口来安装，真是颇奇怪！不过不用理他没关系。 你可以使用上下按键与空格键来进行主要画面的项目挑选， 然后最底下一行有功能键，常见的是『F2』代表继续下一步骤，『F6』代表此画面的详细说明，其它功能键功能则在画面上显示了！ 如上所示，因为所有硬件都没有问题，所以鸟哥选择『No changes』，并且按下『F2』继续下一步； 记得喔，那个『F2』指的是键盘上面数字键上方的功能键喔！

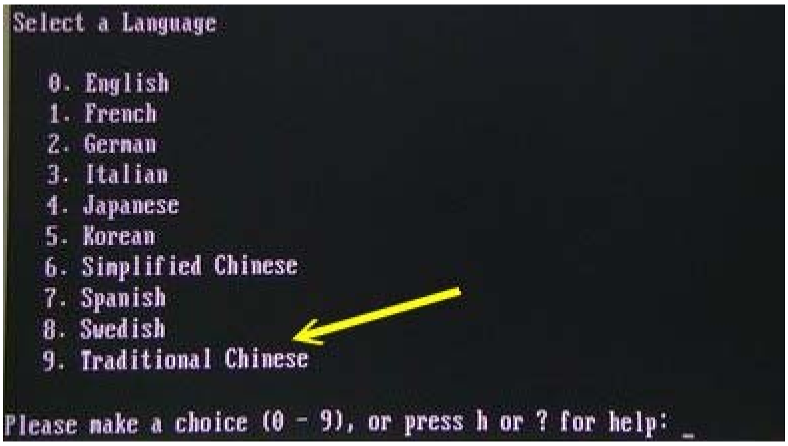


图十三、准备开始安装

在上面这个画面当中，系统会问你要不要进行图形接口的重新测试？既然无法直接进入图形画面，那么再怎么测试基本上都是无效的，而且测试不成功的话，整个画面还会整个花掉～无力～ 所以鸟哥建议你直接使用文字接口来安装即可， 所以请按照上图箭头的指示，按下『F4』即可！

3. 语系选择：

在处理完了一些进入安装画面前的前置作业后，接下来就是要选择画面输出的语系数据了。 如下图所示，Solaris 也是支持多国语系的，下图的第九项就是台湾常用的中文语系了。 不过，因为鸟哥使用的是纯文字的安装（没办法，图形接口无法启用！）， 所以选择 9 还是会出现英文啦！@_@



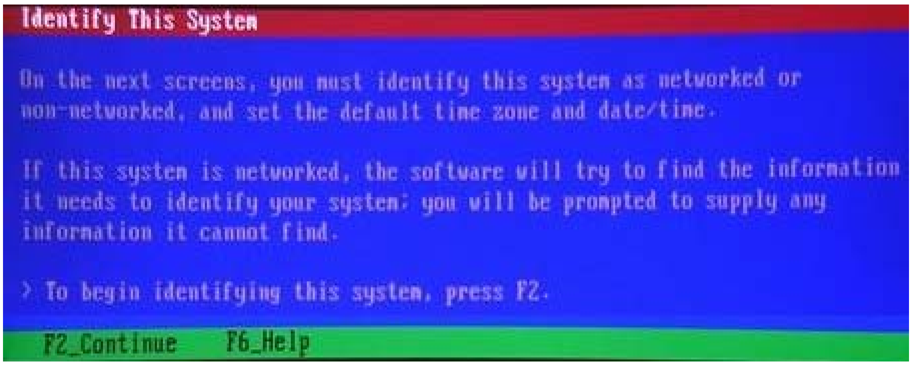
图十四、准备开始安装

上个图示按下 9 之后就会出现如下图示，接下来你最需要察看的就是最底下那一行， 记得那个 F2 不要随便按，看清楚选项后再按吧！ 这里直接按下『F2』就能够进入下个画面。



图十五、准备开始安装

好啦！在下个画面当中在给他按下『 F2 』那个功能键，就可以开始整个 Solaris 预设系统的定义设定画面了！



图十六、准备开始安装

网络相关定义

如果你的安装程序可以捉到您主机上面的网络卡时，那么这个网络定义的部分就会跑出来啦！ 除非你知道你的环境当中网络的相关参数，否则的话，建议你可以直接参考鸟哥的设定值来填写就好了！ 相关的网络功能建议您可以参考底下这一篇：

- http://linux.vbird.org/linux_server/0110network_basic.php

若还有问题，那可就得要仔细的查一查其它的网络书籍了！底下就来仔细的定义网络吧！

---

1. IP 取得的方式：

   既然来到这个画面，这就表示你的网络卡有被捉到啊！请记得，鸟哥的系统上面其实有两张网络卡的， 一张是主机板内建的，一张则是外接的螃蟹卡。不过，内建的网络卡 Solaris 捉不到，所以才使用螃蟹卡啊。 这里要提醒的是：『不要执着于想要立即驱动一些不容易驱动的硬件』，先求有这个咚咚之后，再来想如何驱动的问题啊！ ^_^！在底下的画面当中，你当然应该选择『Yes』啰！

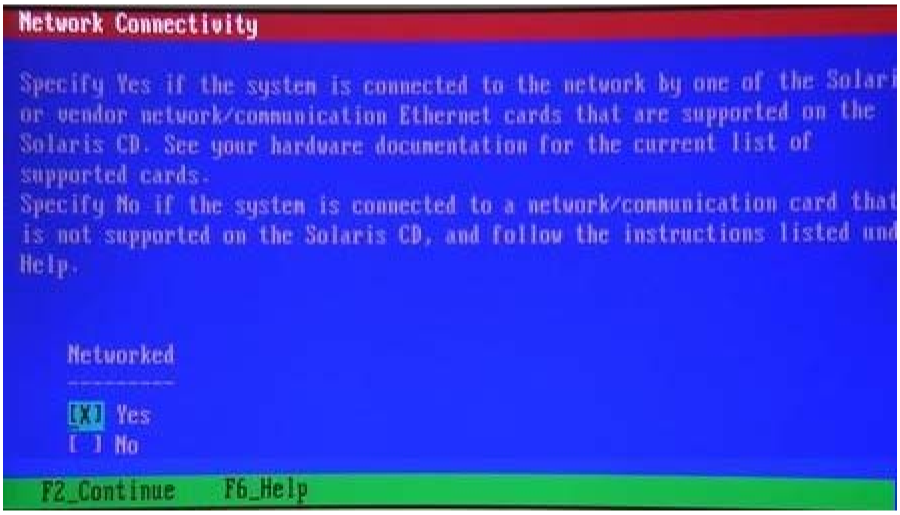


图 17、网络相关参数定义

主机 IP 的取得主要有两种方式，一种是透过类似 IP 分享器（亦即是 NAT 主机）自动取得 IP 的方式， 这种方式就被称为 DHCP (Dynamic Host Configuration Protocol, 动态主机控制协议）。另外一种则是手动给予 IP 的设定数据。由于不知道您是否已经了解 IP 取得的原理， 所以这里建议你直接使用『手动给予』，因此在下列的图示当中，就选择『No』， 不要使用自动取得 IP (DHCP) 吧！



图 18、网络相关参数定义

---

2. 主机名称的设定：

每一部主机都应该要有主机名称的，这样系统才能够以某个名字来启动一些网络的服务。 不过，您的主机名称最好不要与其它 Internet 上面的主机名称相同， 以鸟哥为例，我用我的名字当作是主机名称，我想，会跟鸟哥同样主机名称的其它机器应该是不存在吧！ ^_^！ 鸟哥设定的主机名称是： sun.dm.tsai 。设定完毕后就按下『F2』来继续吧！

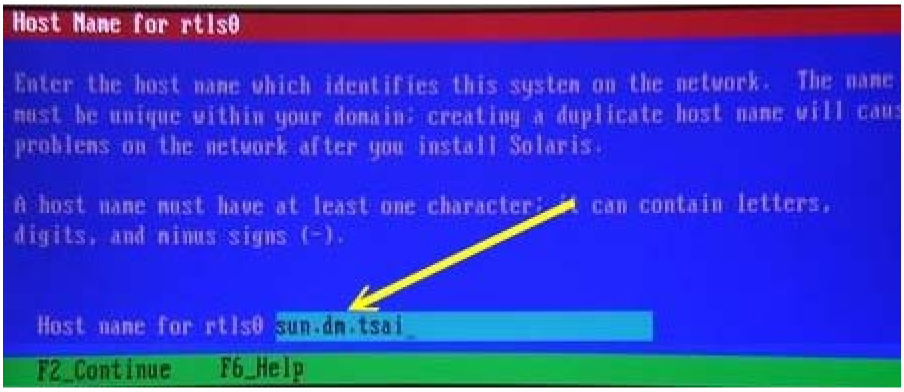


图 19、网络相关参数定义

---

3. IP 参数的给予：

由于在图 18 的地方我们希望手动给予 IP 的相关参数，因此底下这个画面就会出现啦！ 鸟哥在这里预计使用一个 Class C 的网域 (现在这个名词听不懂没有关系，后面网络部分会慢慢介绍)，因此整个想要给予的参数是这样设计的：

- IP：192.168.1.101
- Netmask：255.255.255.0

所以底下的三个图示请分别填写上述的设定值后，然后按下『F2』继续喔！



图 20、网络相关参数定义



图 21、网络相关参数定义



图 22、网络相关参数定义

由于鸟哥的主机环境当中并没有使用到最新的 IPv6 的硬设备，我想，这种网络设备短期内应该不可能会大流行， 所以您可以直接选择『No』就好了。除非您的公司企业内部真的有 IPv6 的设备存在喔!



图 23、网络相关参数定义

4. 路由器（IP 分享器）之 IP 的设定：

每部想要连接到 Internet 上面的主机都需要有路由器的帮助！路由器可以想成是一个数据『转递站』，你可以将他想成是『快递公司』的送货员就是了。我这里假设我的路由器是：192.168.1.2 这部主机，你也可以按照鸟哥的设定来撰写，设定错误也没有关系，反正接下来的几个章节内我们还没有要用到网络啊！等到聊到网络的时候，你就知道如何处理啦！^_^！所以底下的两个画面照样造句的给他完成即可！



图 24、网络相关参数定义



图 25、网络相关参数定义

最后给他看看相关的网络参数有没有设定错误？如下图所示。如果下图显示的信息没有问题的话，按下『F2』就可以继续了。如果发现任何错误，可以按下『F4』来修改喔！



图 26、网络相关参数定义

---

5. 其它网络安全性的规范：

网络相关的数据真的很多，除了上述的重要 IP 参数之外，其实还有所谓的身份控管主机！ 也就是说，你要登入主机时所使用的账号与密码可以由外部主机提供验证的！ 不过我们的主机主要是用来测试用的，尚未使用到外部主机的资源， 所以底下两个选择『Kerberos 及 Name Service』的画面都给他选择『No』即可！



图 27、网络相关参数定义

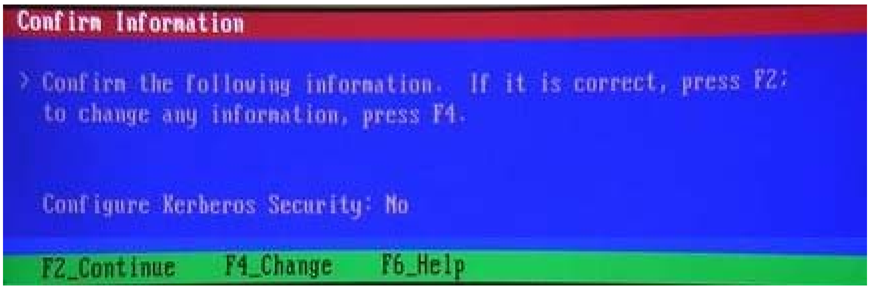


图 28、网络相关参数定义



图 29、网络相关参数定义

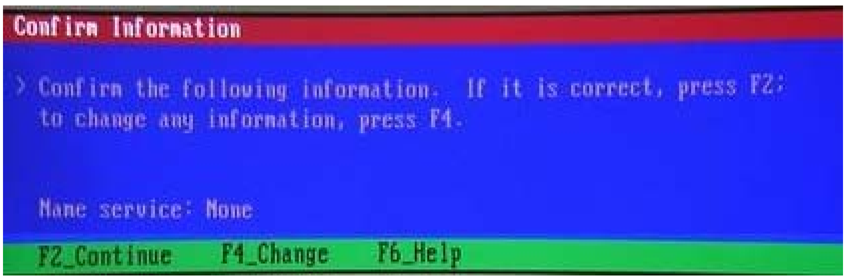


图 30、网络相关参数定义

OK！接下来可以看看时区的设定项目了！

时区的设定与 root 的密码

由于目前 Unix-Like 的操作系统几乎均支持多国语系，例如 Linux 以及这个 Solaris ， 而我们知道地球是圆的，所以台湾白天时，美国是深夜的！所以啦，操作系统当然要提供不同的时区给使用者来选择， 否则在台湾使用美国时间，不是很奇怪吗？您说是吧！所以底下鸟哥是选择：

- 亚洲地区 (Asia)
- 台湾时间
- 目前的时间
- 确认是否有问题！

底下的四个图示你可以依序设定。不过，如果您并非在台湾地区的话， 那么就请依照你自己所在的国度来选择你的时区吧！





图 32、时区的设定项目



图 33、时区的设定项目

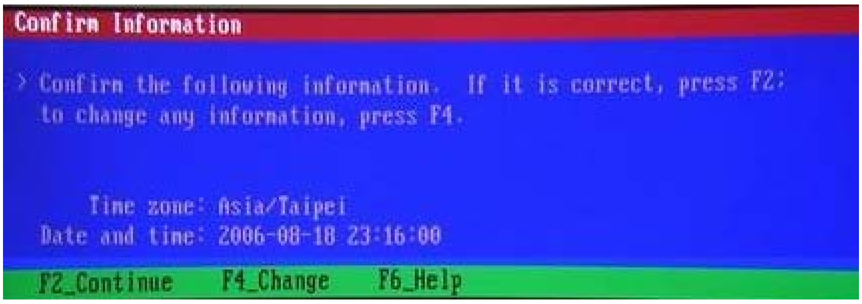


图 34、时区的设定项目

如果每个画面的选择都正确，那么分别给他按下『F2』之后，就能够进入到下个动作！



图 35、系统管理员 root 密码的设定

Unix like 的系统都有个特殊的账号，名称为 root ，这个 root 是『超级使用者，就是系统管理员(administrator, super user)』， 当你有任何系统问题时，都需要这个账号的登入，而当你的系统被入侵时， 被取得的权限也可能是这个账号！因此在这里建议你必须要取个比较好的密码， 会比较保险一点！上图中输入你的密码两次，然后按下『F2』就可以继续下个动作了！

安装的类型

设定完 root 的密码之后，接下来我们可以开始来决定一下这个 Solaris 如何安装到我的系统上！ 这包括全新安装、升级已经其它安装方式等。由于我们是练习用的，之前并没有安装过 Solaris ， 所以底下的画面就不会出现 upgrade （升级） 的选项了！开始玩玩先！

---

1. 安装方式的选择：

如同上面提到的，我们需要的是全新安装，所以底下可以直接使用『F2』那个标准安装 (Standard) 即可。



图 36、安装的类型方式

接下来您可以选择当安装光盘安装完毕时，光驱会自动的退出光盘，这样比较好啦！ 避免我们不晓得啥时候应该要取出光盘啊！所以底下两个画面请依样画葫芦啰！

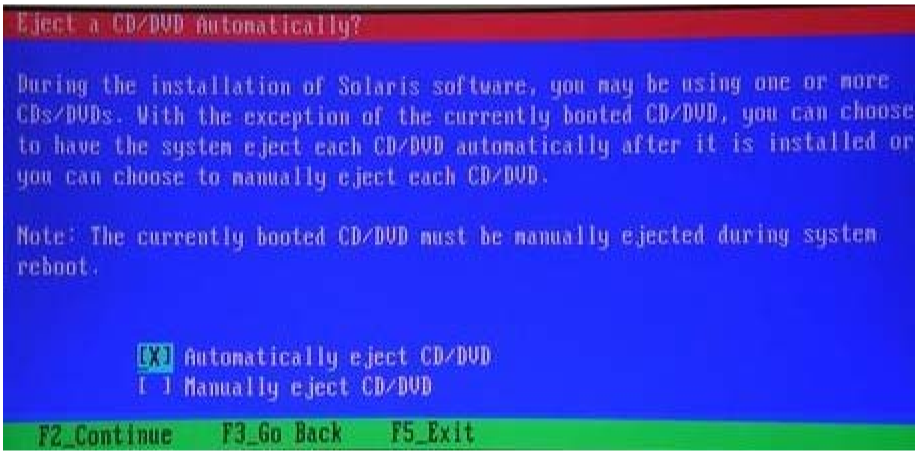


图 37、安装的类型方式



图 38、安装的类型方式

---

2. 同意授权书：

接下来是不可避免的授权书声明啦!你可以自行决定要不要使用 Solaris ，如果确定同意其授权，那按下『F2』也就是了！ ^_^

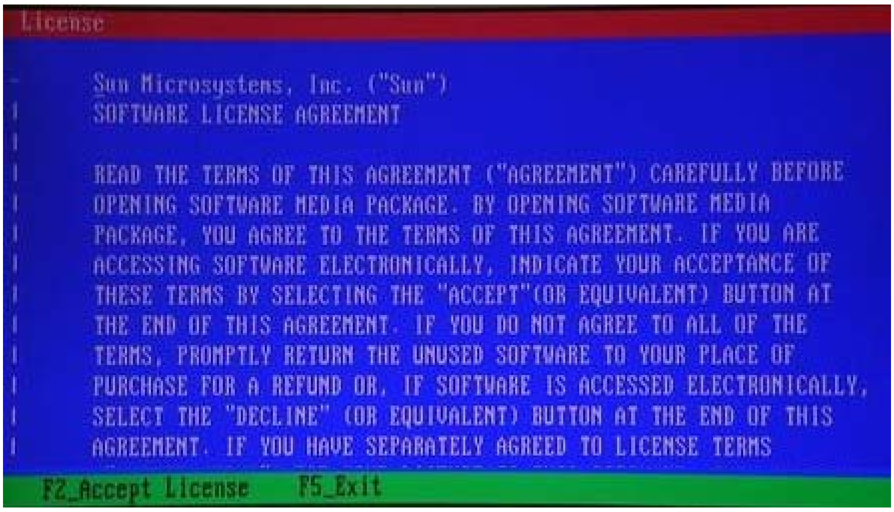


图 39、安装的类型方式

---

3. 主机所在区域选择：

这个与语系及时区有点关系啦！你可以挑选您主机所在的区域， 然后选择预设语系，可以方便您加入一些额外语系的支持喔！



图 40、安装的类型方式

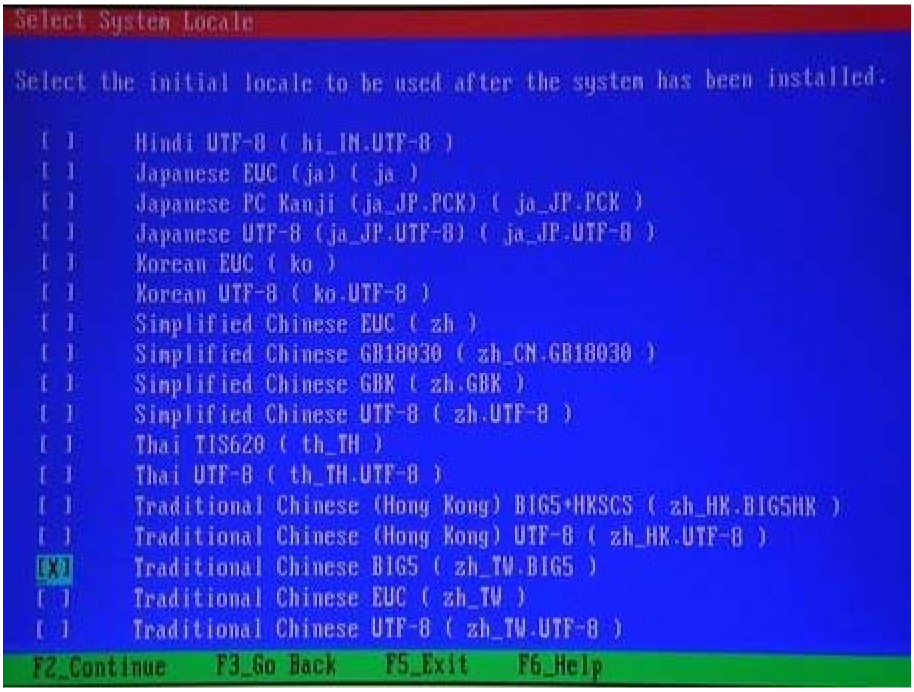


图 41、安装的类型方式

---

4. 额外产品的安装：

Solaris 额外支持类似 Java 程序语言执行的环境，如下图所示， 由于我们目前是想要摸索 Solaris ，还没有玩到 Java 这类的程序啦！所以鸟哥这里两个都没有选择， 直接按下『F2』给他继续去！



图 42、安装的类型方式



图 43、安装的类型方式

搞定这些基本数据后，嘿嘿！接下来进入最麻烦的硬盘分割啦！

磁盘分割

接下来就是那个麻烦到快要爆的磁盘分割了！请记得，在 Solaris 的磁盘代号是如何啊？ 忘记的话先回去前面的章节翻一翻先！这很重要喔！不要忘记了！在鸟哥的这个例子当中， 我是使用 40GB 的硬盘仅取 20GB 出来玩 Solaris ，同时将原本硬盘的数据通通删除喔！ 如果你要做多重开机的话，那么底下的一些步骤需要特别留意！别通通照着做！要仔细的看各项说明喔！ ^_^

---

1. 选择安装的软件套件内容：

   在 Solaris 内共有几个常用的软件套件安装内容，如下图所示，他们的内容分别是：

   -
   - Reduced Networking Core System Support：
     算是 Solaris 的最小安装吧，里面仅有开机所需数据以及有限的网络服务功能， 但依旧可提供多人的文字接口终端机以及系统的管理工具等；

   - Core System Support：
     包含 Solaris 的开机数据以及最小化的 Solaris 设定信息等，但一些服务器常见的应用软件则没有安装。在网络功能方面主要含有 telnet, ftp, NFS, NIS 等服务，桌面方面则包含有 Sun 自家的 Common Desktop Environment (CDE) 的桌面环境，不过桌面的应用软件则没有安装，也缺乏很多的在线文件数据 (online manual pages)；

   - End User System Support：
     包含了 Core System Support 的内容外， 还有常见的桌面应用软件等功能；

   - Developer System Support：
     包含了 End User System Support 的所有内容外，主要还提供了程序与系统发展者所需要的函式库与相关数据 (library, include file, onlin manual pages) ，还有一些发展维护的工具等等；

- Entire Distribution：
  包含了 Developer System Support 的所有数据外， 也添加了其它主机所需要的相关软件，这也是 Solaris 预设选择的安装软件方式；至于 Entire Distribution plus OEM support 则更添加其它的 OEM (Original Equipment Manufacturer, 原始设备制造) 所需要的硬件驱动程序等数据。

因为我们是首次接触到 Solaris 这个玩意儿，想要完整的玩玩这个咚咚，最好还是完整安装比较妥当， 因此你可以直接选择『Entire Distribution』的安装方式即可。

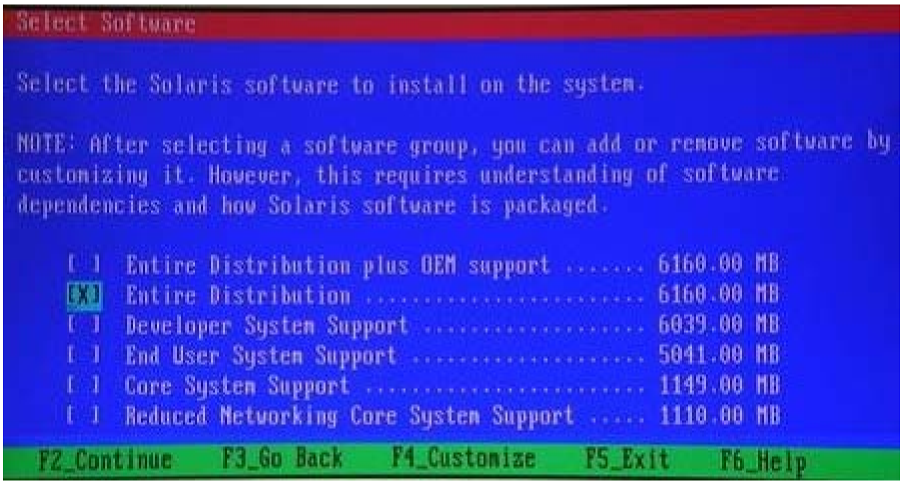


图 44、磁盘分割

---

2. Solaris 整体 partition 的容量选择：

因为鸟哥选择了『Entire Distribution』的软件安装方式，总共至少也需要 6GB 的硬盘空间， 而鸟哥预计是拿 20GB 来玩这个 Solaris 。但如同前面提到的，Solaris 的磁盘分割与一般 Windows/Linux 不太相同，他仅能使用 Primary Partition 来进行分割动作， 因此底下的动作要特别小心啊！首先如下图所示，我们要选择某一颗硬盘来作为 Solaris 的分割区， 因为鸟哥仅有一颗硬盘，不过在安装之前这颗硬盘显示的结果却是怪怪的，不理他， 直接使用『F4』按键来选择 Solaris 所需要的全部容量！

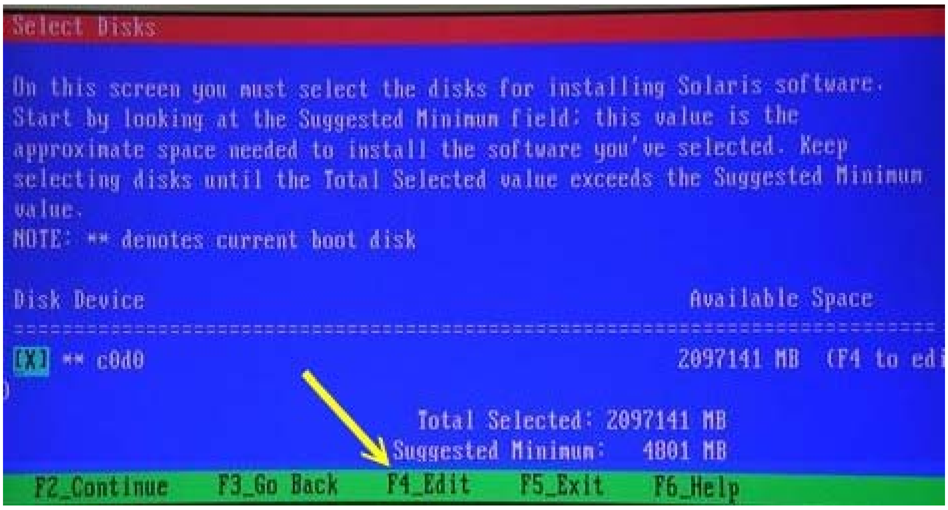


图 45、磁盘分割

我们所要进行的工作是分割，所以在下图当然选择『Edit Fdisk partitions』项目，按下『F2』继续。



图 46、磁盘分割

呐！看到了吧！在 x86 的硬盘里面第一个扇区 (sector) 由于最多仅能容纳 4 笔 partition 的数据，所以在底下的这个图标当然显示 1~4 号啰!由于鸟哥这颗硬盘内所有的数据都不要了，所以将光标以上下按键移动到 Other 及 SOLARIS 的地方后，按下『F3』来删除这些 partition 的数据！

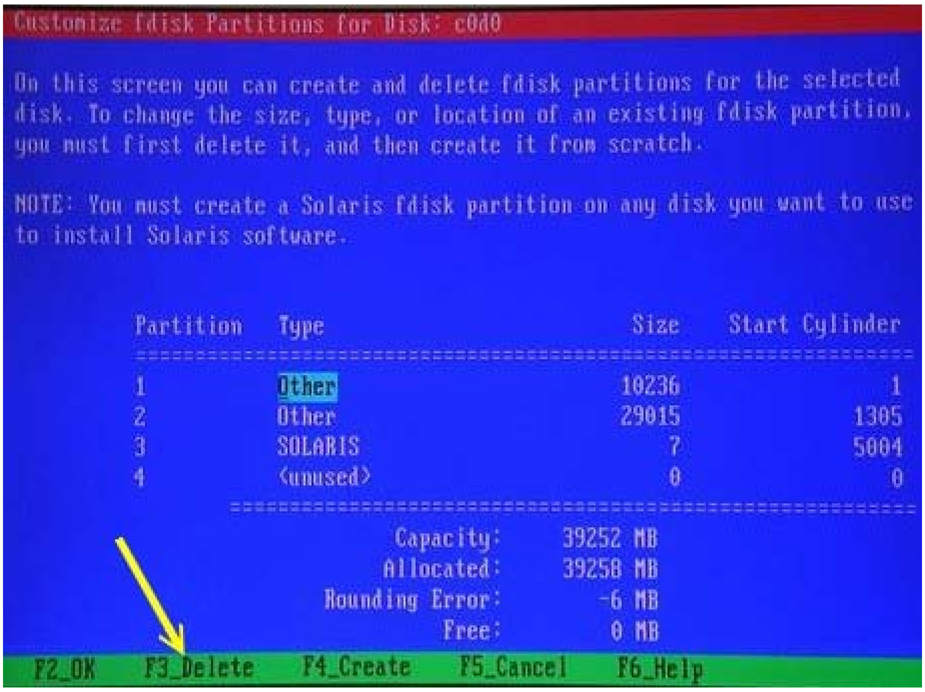


图 47、磁盘分割

将上图的所有分割信息都删除之后，就会得到如下图的样子，终于看到正确的硬盘空间 (约 40GB) 啦！ 鸟哥预计利用第一块 partition ，所以将光标移动到 partition 1 的地方，然后按下『F4』来建立 Solaris 所需要的磁盘空间；

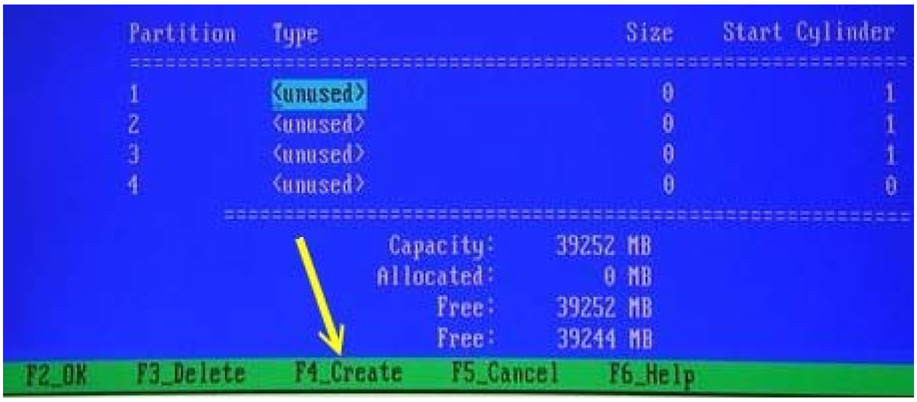


图 48、磁盘分割

注意看下图啊！实际的分割信息其实记录的是『Cylinder』，不过我们人类对于磁柱的计算方法总是不太了解， 尤其每颗硬盘的磁柱大小均不相同，因此可以直接在『Partition size(MB)』的地方填入我们所需要的 20000 MB (约为 20GB 即可)，然后再按下『F2』就可以建立 Solaris 所需要的磁盘空间啦！ 不过此时其实仅只是规划出 Solaris 用的磁盘是由哪里到哪里，尚未处理目录树的挂载数据啦！



图 49、磁盘分割

看一看下图的的输出结果，在 partition 1 的地方确实是 Solaris 的分割区，并且含有 20010 MB 的容量， 如果没有错的话，就可以输入『F2』回到硬盘选择的画面啰！

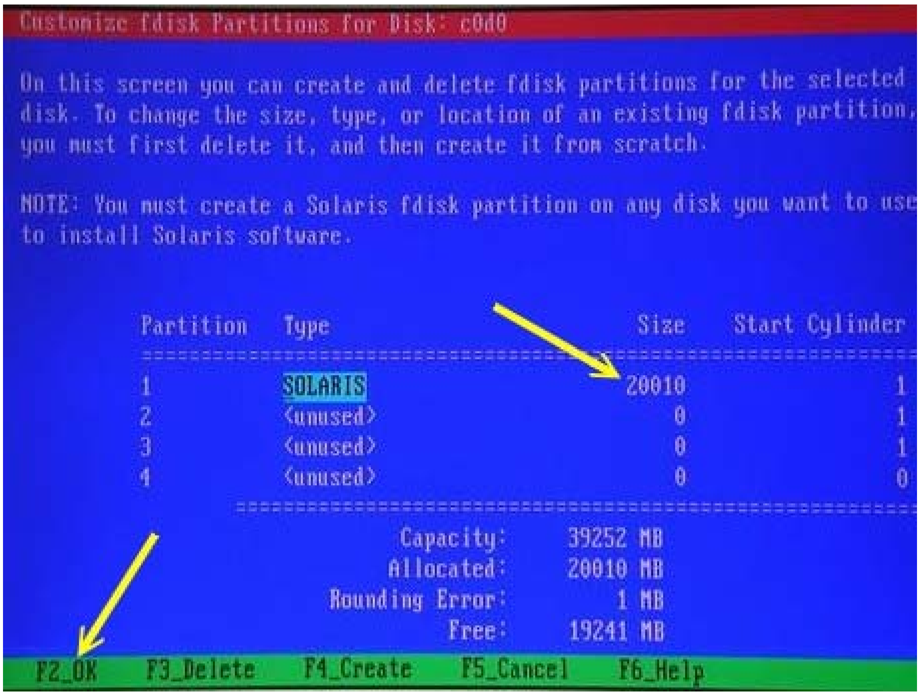


图 50、磁盘分割

下图如果确定该颗硬盘的容量大小 OK 的话，按下『F2』之后，准备要将 Solaris 的分割区块再与 Solaris 的档案系统挂载上来啰！

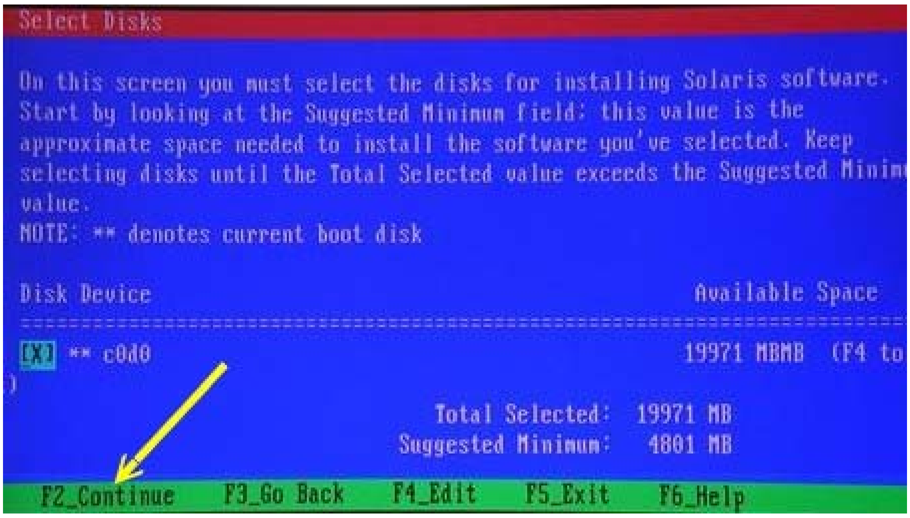


图 51、磁盘分割

---

3. Solaris partition 内细部的分割信息与挂载：

上一个小动作在建立 Solaris 全部数据所需要的磁盘空间，在这里我们必须要将这些磁盘空间再进行细部的分割， 以使这些磁盘空间可以与 Solaris 的档案系统 (filesystem) 连结上才行！如下图所示，将档案系统与磁盘分割连结上的方式可以选择：

1. 让安装程序自动分配 (Auto Layout)；
2. 手动分配 (Manual Layout)。

我们先按下『F2』选择一下自动分配会是什么情况？

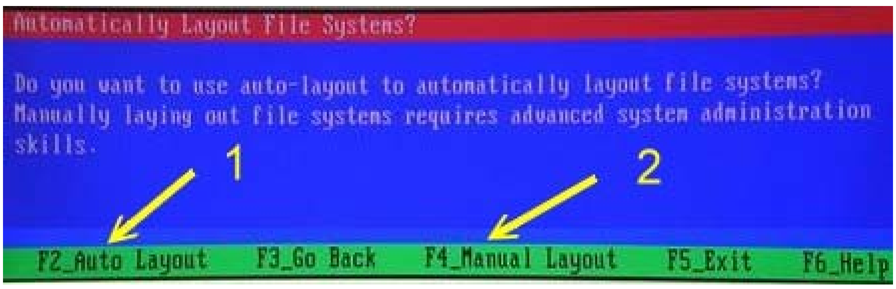


图 52、磁盘分割

在上图按下『F2』之后会出现下图，Solaris 建议需要独立出来的目录有 /，/opt，/usr，/var，swap 等等， 其中 / 与 Swap 是『必要的！』，知道这些之后， 我们可以选择『F5』取消自动分配，回到图 52 的地方重新按下『F4』来手动的建立比较好啦！

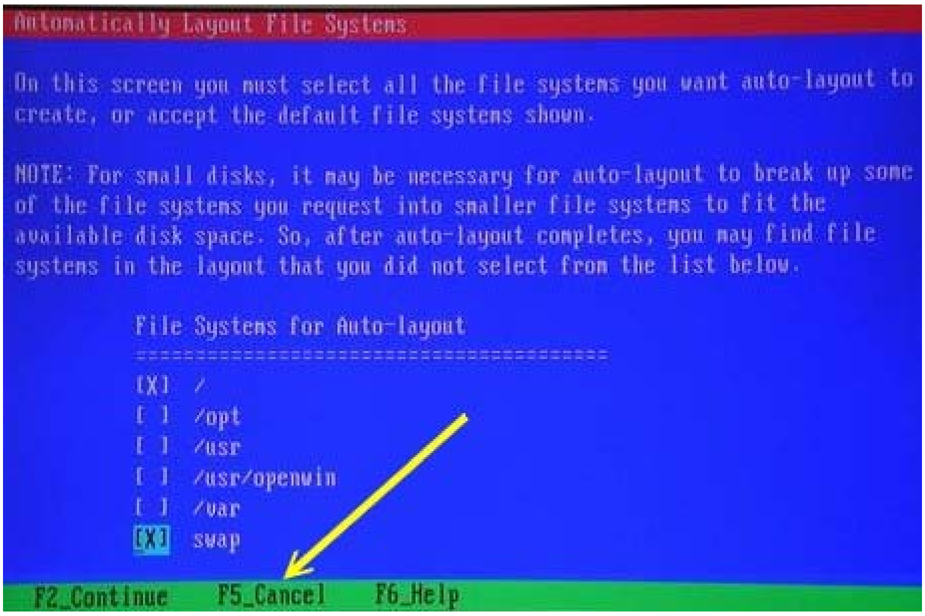


图 53、磁盘分割

在图 52 的地方按下『F4』就会出现下图，仔细看一下那个磁盘代号『c0d0s2』，还记得我们在开始安装之前的分割介绍吧？ 没错！第二号 (s2) 分割主要是记录整个 Solaris 磁盘分割槽的整体记录，所以这时才会出现这个画面啊！ 接下来我们就可以使用『F4』来自动配置目录与分割的挂载内容了！

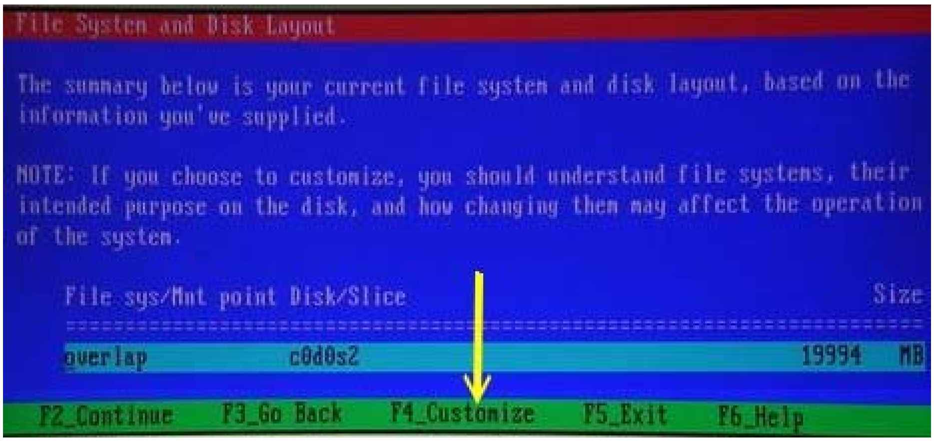


图 54、磁盘分割

嘿嘿！如下图所示，我们可以利用的分割代号分别是 c0d0s0, c0d0s1.....c0d0s7，其中 c0d0s2 是 overlap， 你不能去更动他的！你可以使用上下键去更动 Mount Point (挂载点)，这个 Mount Point 就是 Solaris 目录树的相关目录啦，如前一个画面谈到的，必要的目录树是『 / 与 swap 』啦！那个 swap 是虚拟内存的意思。 同时注意到下图最底部有个『Free』的数据，你最好将该空间全部给他分配完毕啊！

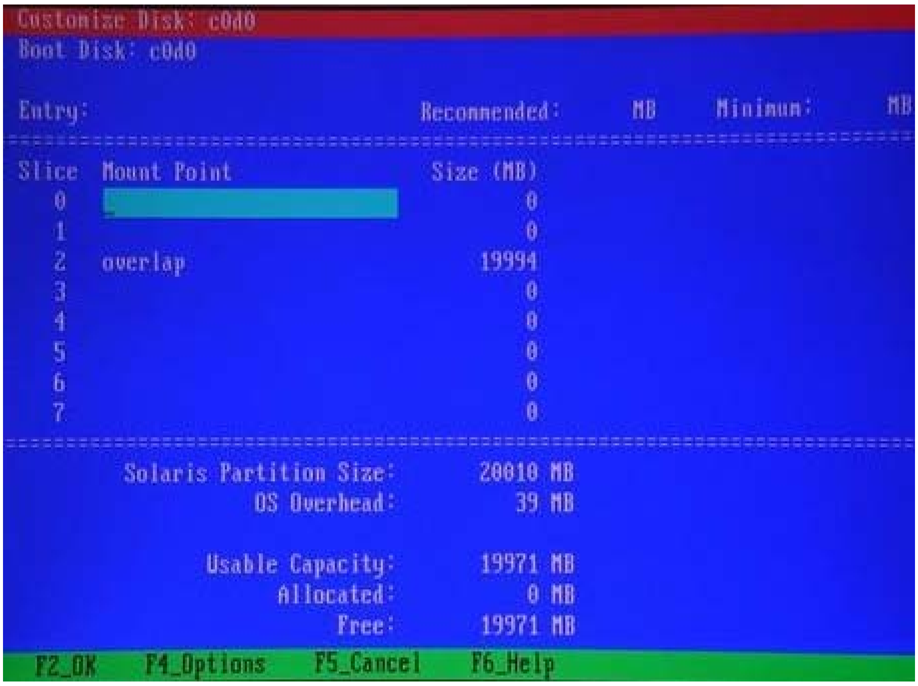


图 55、磁盘分割

如下图所示，当你在 Slice 0 的地方输入根目录（/），然后将光标移动到 Size 的地方， 结果在下图箭头显示（3）的地方就会出现一个建议值，该建议值你可以自行参考， 不过鸟哥建议你最好还是考虑自己的环境来设定你的容量啊！

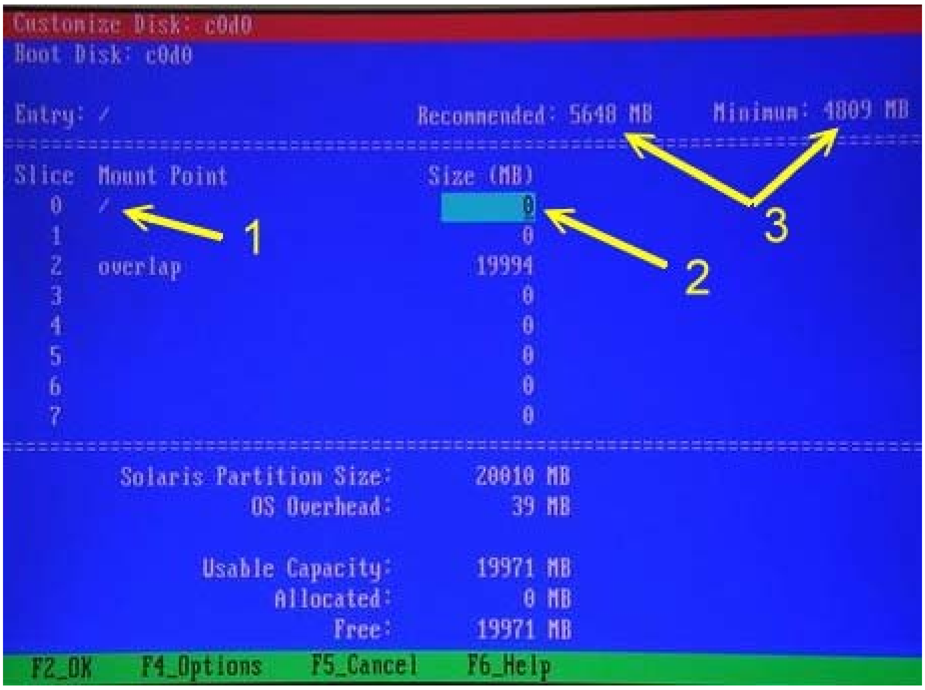


图 56、磁盘分割

如下图所示，鸟哥最终决定制作出 /, swap, /home, /var 这四个 Solaris 内的分割区， 同时将 20GB 的空间通通分配给这四个分割槽，所以您会发现下图最底部的『Free』是被用光光的！ 其中只要记得根目录『 / 』不要给太少，而 swap 不要超过 512 MB 就可以了！ 如果没有问题的话，那就按下『F2』来继续吧！

不过，在 Solaris 内 /home 有特殊用途，并不是使用者预设的家目录， 一般建议您应该要建立 partition 对应在 /export 或者是 /export/home 这两个目录底下喔！ 这个问题刚接触 Solaris 的朋友最容易犯！当然鸟哥也不例外！ @_@！ 那如果你已经安装了 /home 了，可以参考本章文末的延伸阅读来修订喔。

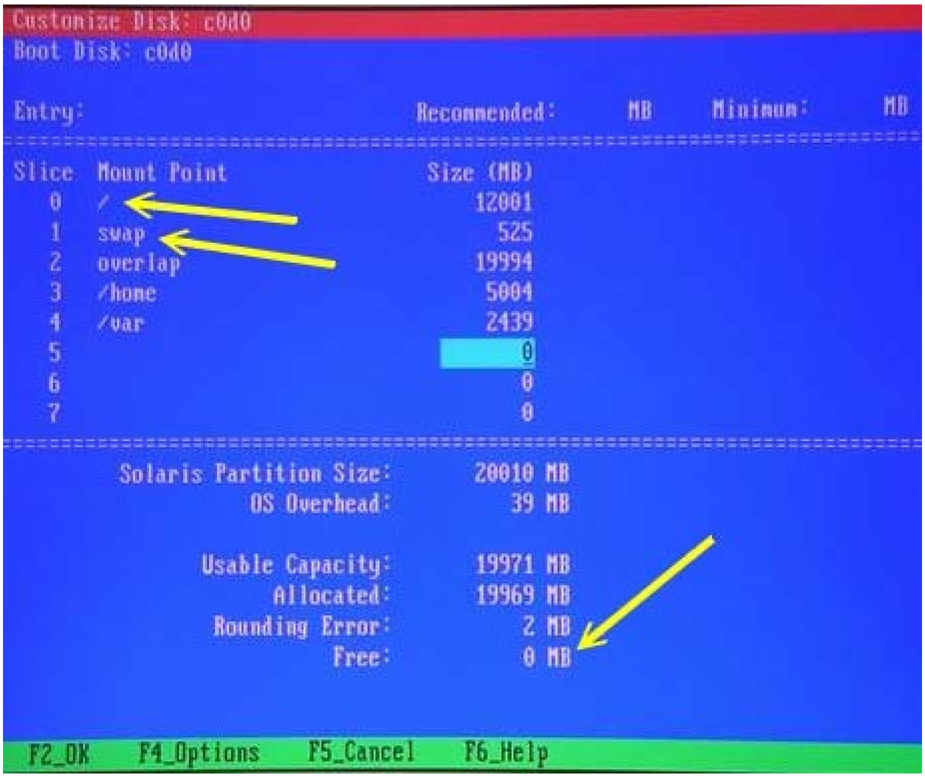


图 57、磁盘分割

一切 OK 的话就会出现如下图 58 所示的 Solaris 磁盘利用区内的分割槽了，对照着图 54 的输出结果来看， 我们已经制作出 /, swap, /var/, /home 了，所以按下『F2』安装程序就能够继续啰！



图 58、磁盘分割

Solaris 还允许我们直接利用网络磁盘，但是目前我们没有啊！所以下图直接按下『F2』就好啦！



图 59、磁盘分割

最后的结果如下图所示，再仔细的看一下有没有错误的地方，如果没有错误的话， 那就『F2』给他开始进行格式化与安装了吧！Oh, Ya!



图 60、磁盘分割

开始实际安装软件

好啦！分割也作完了，终于可以进行安装啦！安装的过程就是一再地抽换光盘就是了！

---

1. 第一片光盘的安装：

在上个图示按下『F2』后，闪过一些格式化的画面后， 就会进入到如下图的安装流程画面，这个画面在第一片光盘安装完毕后就会进入下个画面；

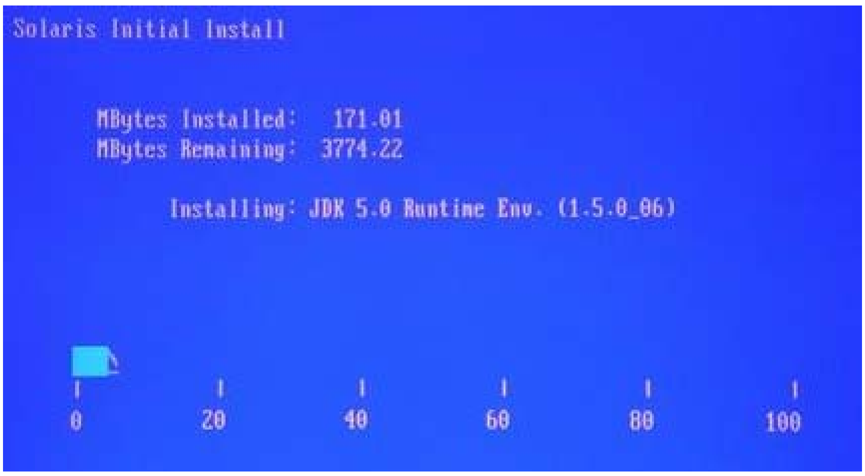


图 61、开始安装软件的流程

第一片安装完毕，会出现如下的画面，并且会告知使用者系统将重新开机喔！ 此时请将第一片光盘取出，否则又会再次的进入安装程序！注意注意！

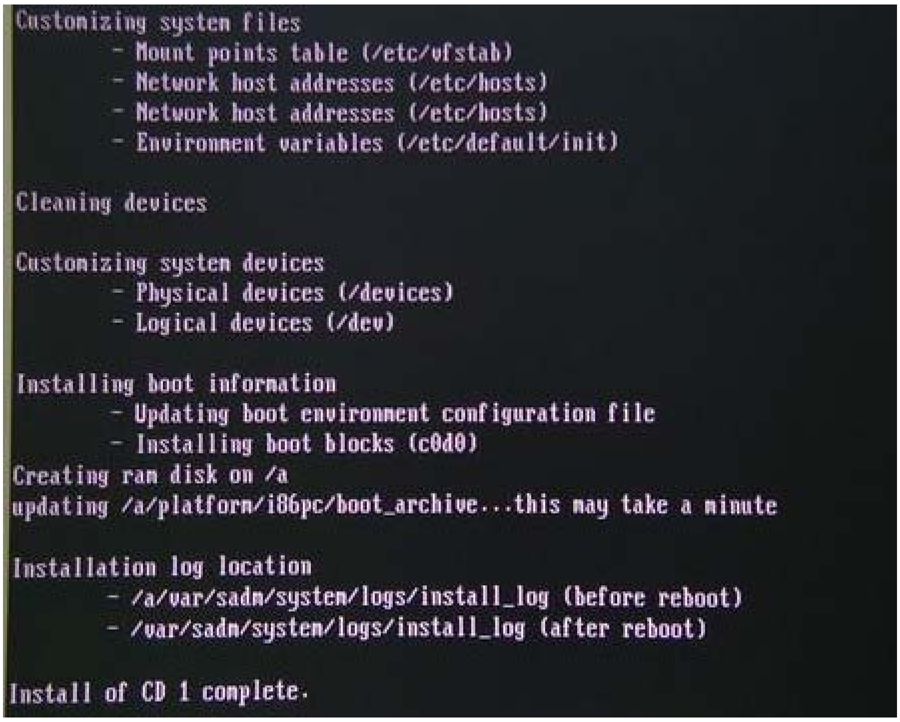


图 62、开始安装软件的流程

---

2. 重新开机后可能遇到的问题：

看吧！立刻重新开机了！而且开机会显示如下画面的选单， 在该选单上头选择第一项开机即可。第二项开机是当 Solaris 系统出问题时的『安全模式』啦！ 另外，在此时『您最好已经将第二片光盘放入光驱当中啦！』， 如此一来就能够继续安装后续的光盘啊！

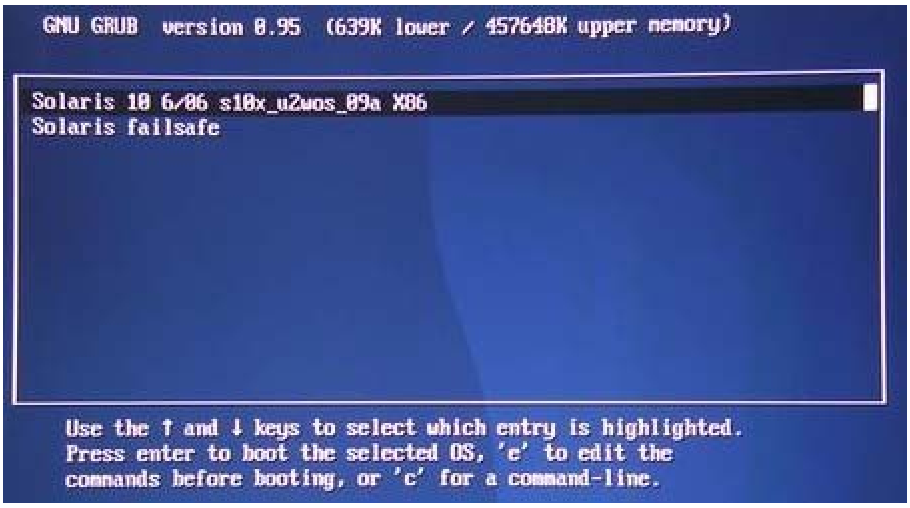


图 63、开始安装软件的流程

在屏幕上闪过一些信息后，系统会问你要不要使用 NFS version 4 的字样，我们尚未使用到网络功能， 所以这里输入 no 或者是直接按下『Enter』也可以！如下图所示：

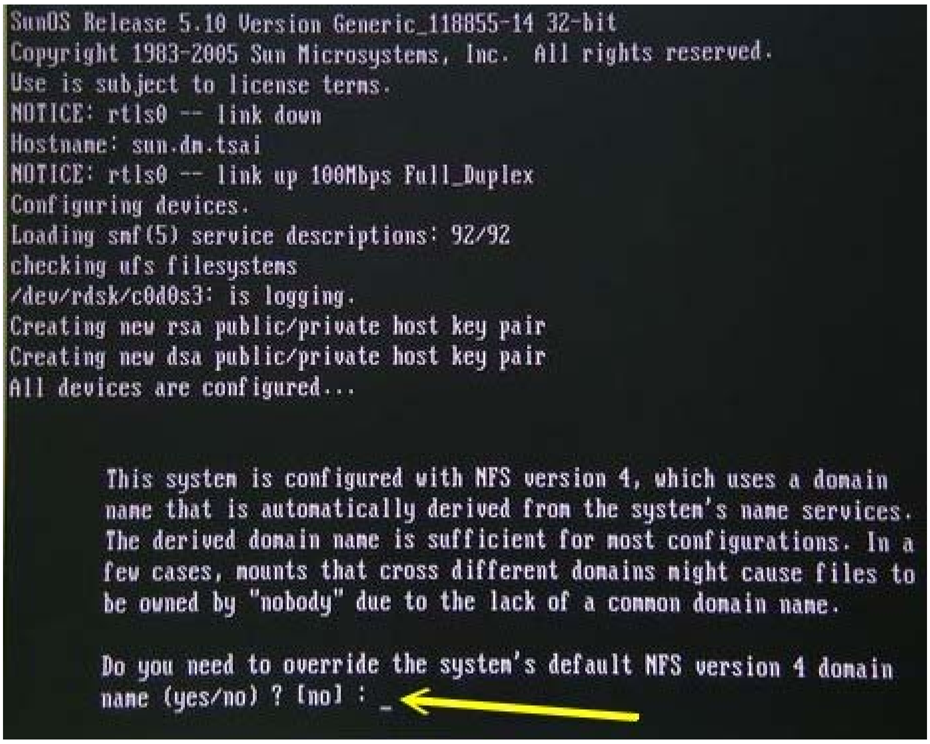


图 64、开始安装软件的流程

还记得我们在安装的过程当中曾经发生无法进入图形接口的问题吧？没错，这个地方原本应该要使用图形接口的， 不过由于安装程序捉不到显示相关的信息，所以会出现一个警告窗口。在你的环境里面可能不会出现这个问题啦！ 在鸟哥这次的测试当中则有出现这个状况。请在下图的地方输入刚刚我们上头所建立的 root 的密码即可；

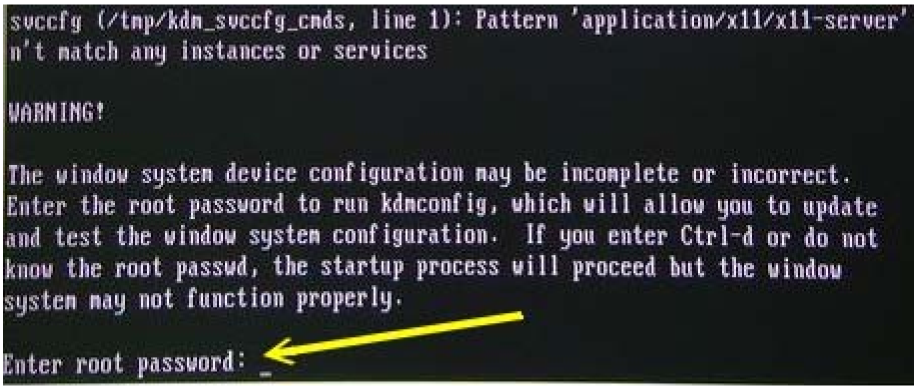


图 65、开始安装软件的流程

接下来则是要使用者重新设定好图形接口的一个程序，称为『kdmconfig』这个程序， 仔细看一看下图的设定值，咦！又是没有问题！伤脑筋～所以还是直接按下『F2』即可;

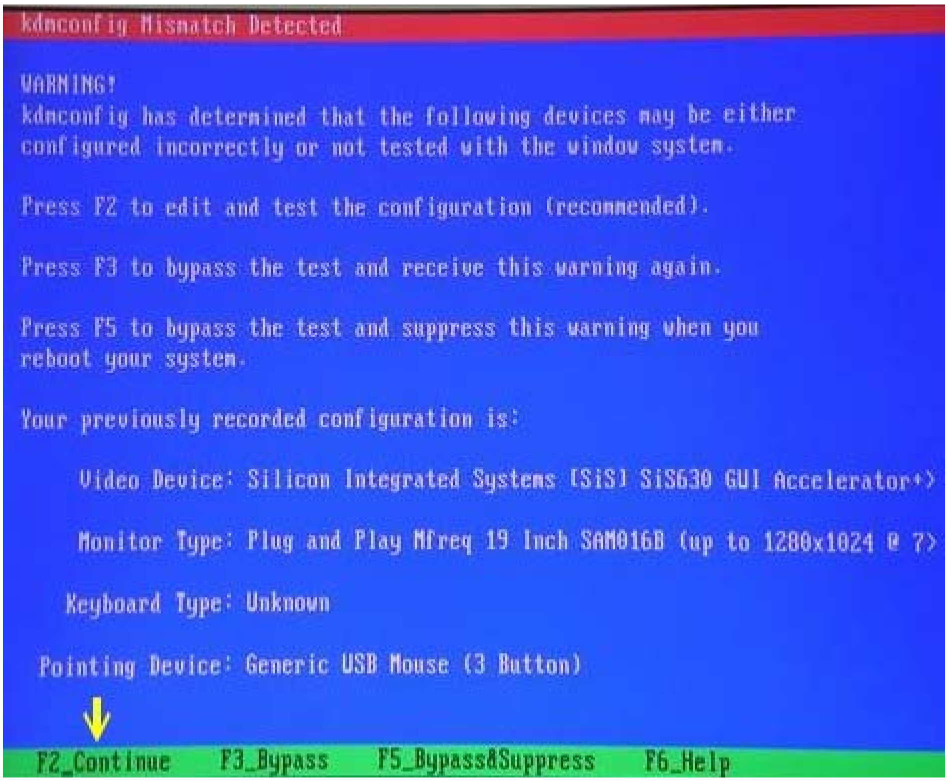


图 66、开始安装软件的流程

既然知道我们的安装程序无法捉到相关的显示信息，那么直接选择『No changes needed -Test/Save and Exit』项目， 然后按下『F2』即可;

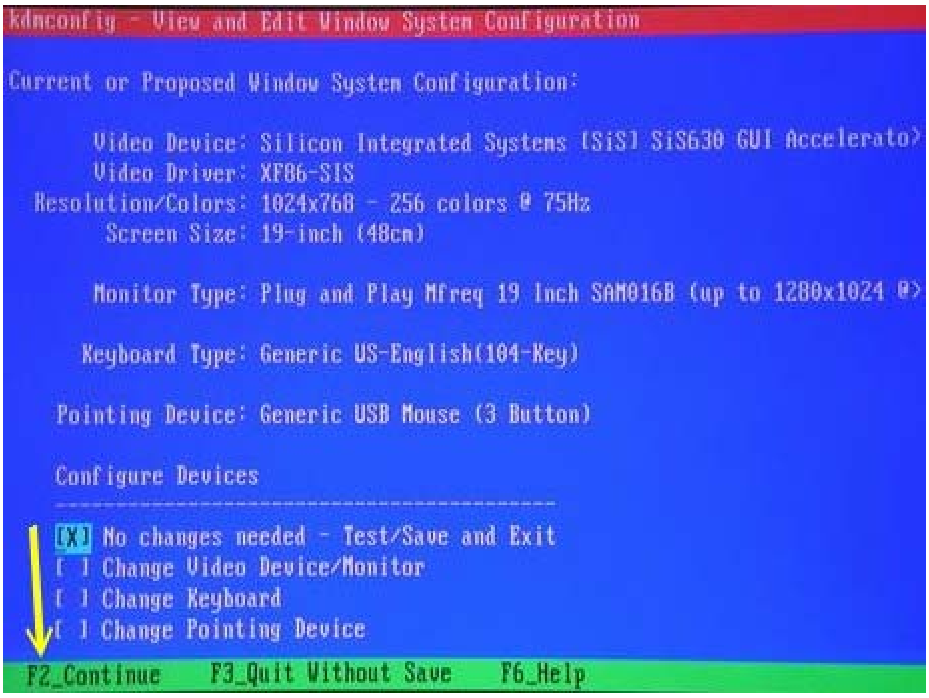


图 67、开始安装软件的流程

接下来可就重要了，如下图所示，请千万按下『F4』啊！不要测试 (Test)， 因为本来安装程序就捉不到显示的相关信息，如果按下『F2』的话，就会出现图 69 的错误信息；

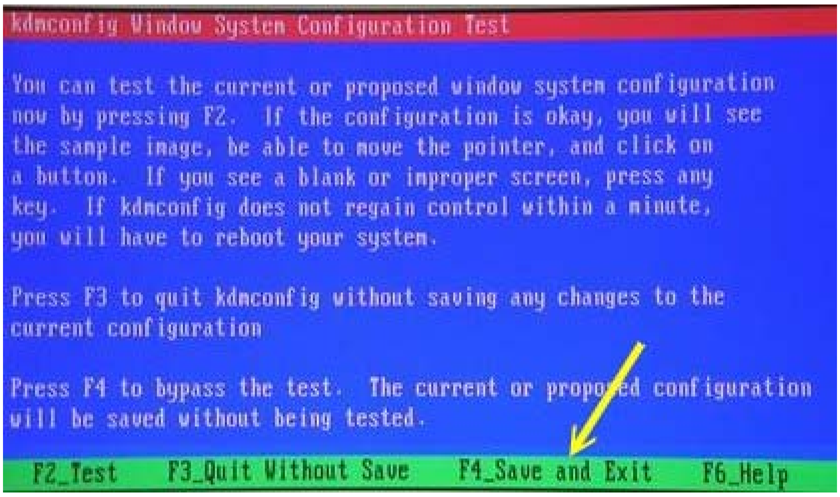


图 68、开始安装软件的流程

如果你在图 68 按下『F2』那你的屏幕就会变成如同下面图示的这附德行～整个画面给他错误掉了 ～伤脑筋～你当然可以继续安装的动作， 不过就是屏幕乱了点，很难看懂安装的过程就是了。因此记得在图 68 的地方要按『F4』才行喔！



图 69、开始安装软件的流程

---

3. 后续光盘片的安装：

顺利的略过 X Window 的侦测之后，接下来就要开始进行第二块光盘的安装了，在下图箭头处输入『Enter』， 并且确认光驱内有第二张光盘片存在，就能够进入安装进度的画面；

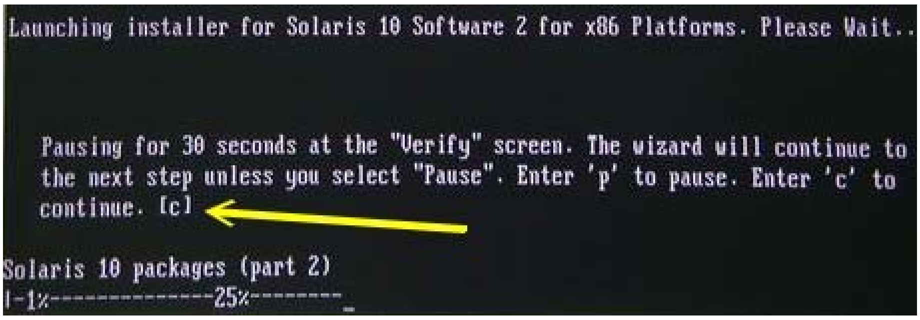


图 70、开始安装软件的流程

在安装完第二片光盘后，第三片之后的安装流程都差不多相同， 你必须要在下图箭头处先按『Enter』然后选择『1. CD/DVD』的项目， 然后抽换一下光盘，如下图所示；

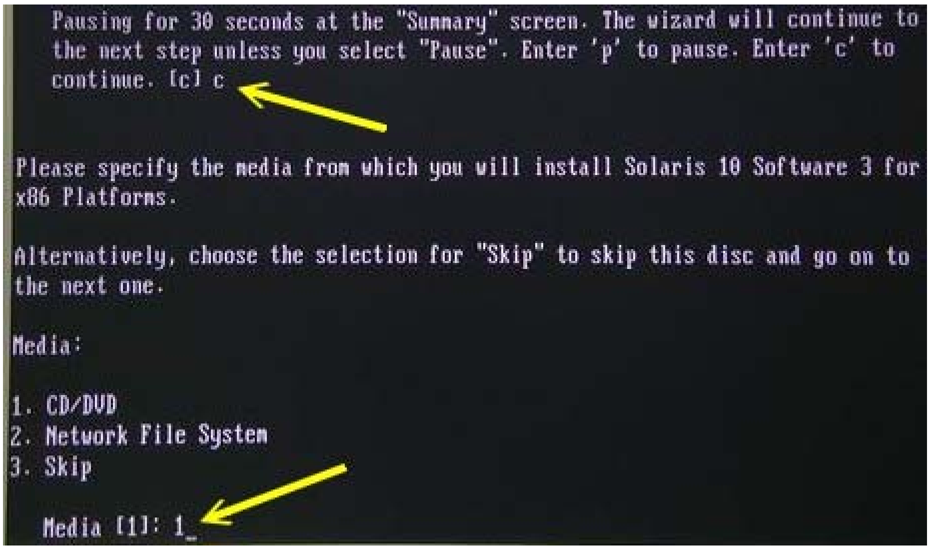


图 71、开始安装软件的流程

确定我们是有光盘的，所以在下图直接按下『Enter』即可；

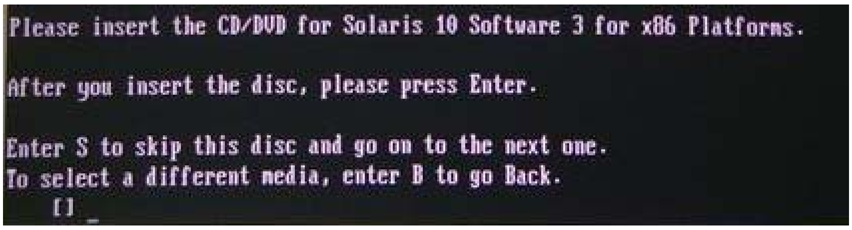


图 72、开始安装软件的流程

如下图所示，按下『1』，亦即立即安装 (Install Now) 就能够继续安装了！

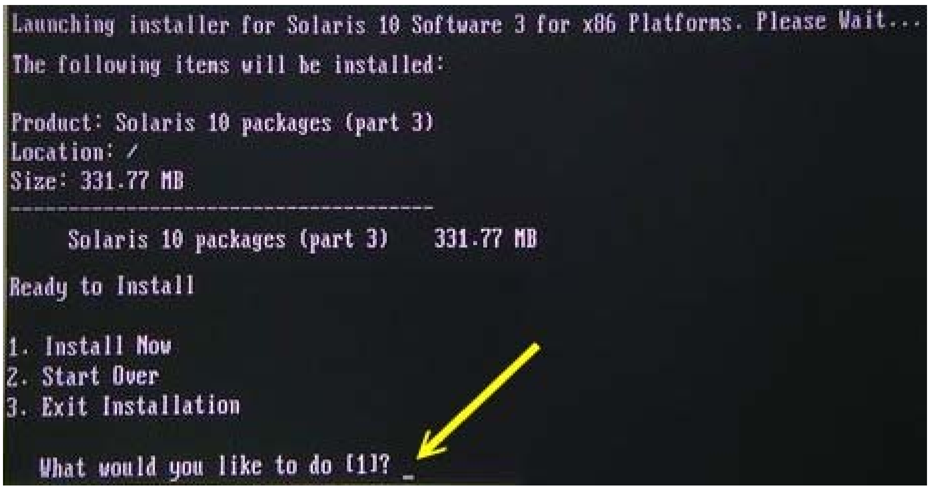


图 73、开始安装软件的流程

重复上面的流程将六片光盘通通安装完毕后，差不多也花掉了鸟哥两个多小时快要三小时的时间！

哇！真累～ 最后看到如下的画面后，取出你的光盘片，然后按下『Enter』就能够重新开机啦！真是好啊！



图 74、开始安装软件的流程

整个安装的程序就到这里啦！也就是说，你已经将 Solaris 安装完毕了！嘿嘿！不容易啊不容易！ 鸟哥第一次安装的时候，竟然花掉了 3 个小时在安装，以鸟哥这部主机的配备来说， 安装 3 个小时确实是花去了太多的时间了！另外，安装的接口确实不是很理想， 如果 Solaris 想要打入一般使用者的市场的话，那么整个安装的接口应该要更加的流畅才行！ 这方面建议可以参考 Red Hat 或者是 SuSE 的安装接口，如果能够更容易安装的话， 对于 Solaris 的推广应该会比较容易吧！

进入 Solaris 系统

如果一切顺利的话，那么你应该可以看到如下的开机画面：

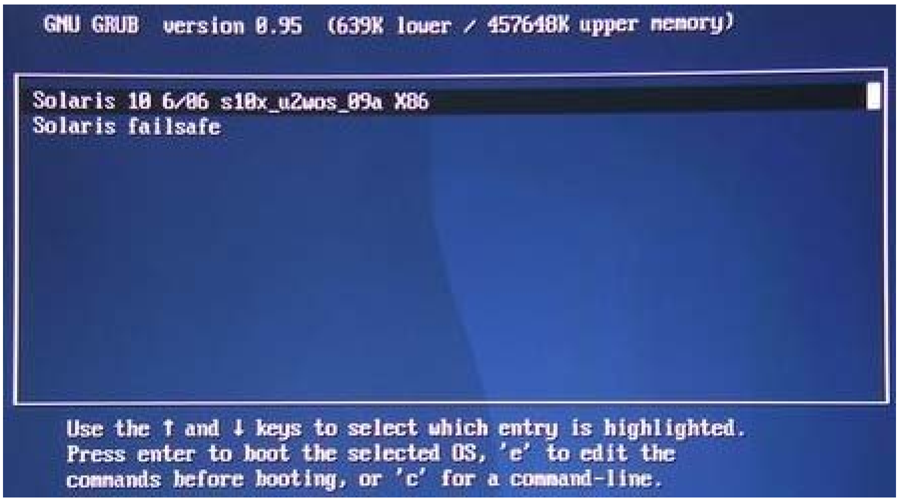


图 75、重新开机进入开机选单

在这个开机选单当中，同样的选择第一项来正常的开机，然而由于我们在安装的过程中本来就没有办法启动 X 窗口画面， 所以会出现如下的登入画面：

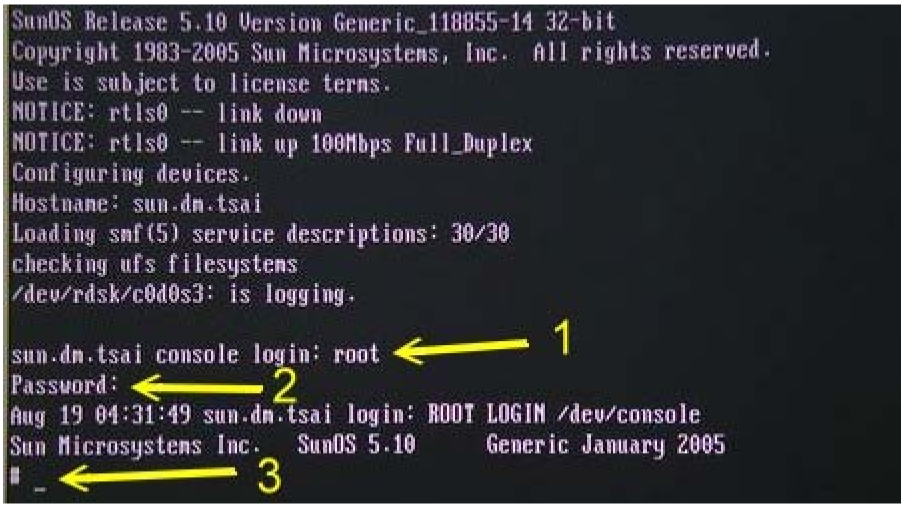


图 76、等待登入的画面

在上面的画面当中，我们可以先使用系统管理员，亦即是 root 这个账号来登入系统， 分别在箭头 1 的地方输入 root ，箭头 2 的地方输入密码， 要注意你输入密码的时候屏幕不会出现任何讯息， 不要以为键盘坏掉了！^_^！最后系统显示系统信息，并且提供箭头 3 的让你打指令的一个提示字符与光标！ 呵呵！你已经顺利的入 Solaris 了啦！先来高兴一下！

不过高兴没有多久就陷入一片苦海当中，因为系统出现如下的画面：

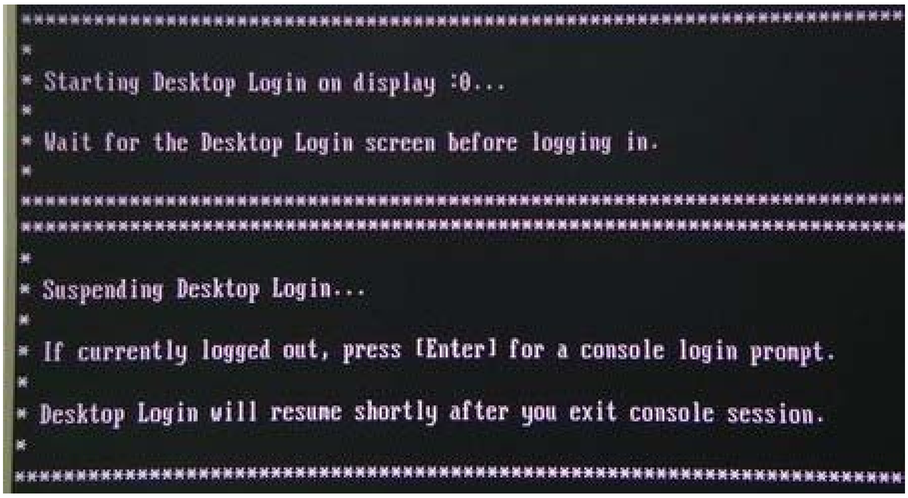


图 77、无法取得 X Window 时显示的错误讯息

唉！因为我们无法使用 X Window 嘛！那如果真的想要使用 X Window 该如何是好？

初次进入 Solaris 作的手脚

如果你的 X 无法启动的话，偏偏在安装的过程当中又可以找到你的显示卡相关参数， 只是无法启动而已，那么你可以到底下这个目录来动作看看：(注意，如果你一直无法进入 X ， 那么先按一次『Enter』取得那个 # 符号的提示字符，再开始底下的指令输入)

```
# cd /etc/X11
```

```
# ls -a
.xorg.conf
....其它省略....

# mv .xorg.conf xorg.conf
```

那个档案 xorg.conf 是 X 的主要设定文件，可能由于安装程序的缘故，所以被取名为小数点开头的档名，而这个小数点开头的档名就是我们一般谈到的『隐藏档』， 所以我们使用 mv 这个指令来更改文件名成为设定档的档名，档案更改完毕之后，你可以这样输入：

```
# gdm
```

那个 gdm 是个指令，他可以使用另一个图形显示管理的工具来启动我们的 X ， 由于鸟哥的主机无法使用预设的 X 登入服务来登入我们的窗口画面，所以只好先使用这个 gdm 先作为一个替代方案啦！到这里为止，你应该就能够登入你的窗口画面啰！如果你的窗口一直怪怪的， 那么在任何窗口画面的地方，直接按下『[Ctrl]+[Alt]+[backspace]』这三个组合按键， 那个[backspace]就是退格键啦，那你的 X 就会关闭啰！ ^_^

进入 CDE 环境

既然要玩 Solaris 当然就得要玩一下 Sun 的预设桌面窗口，亦即是 Common Desktop Environment, CDE 这个咚咚啰～所以当你的窗口终于出现如下画面时，你可以自行挑选 X 喔！

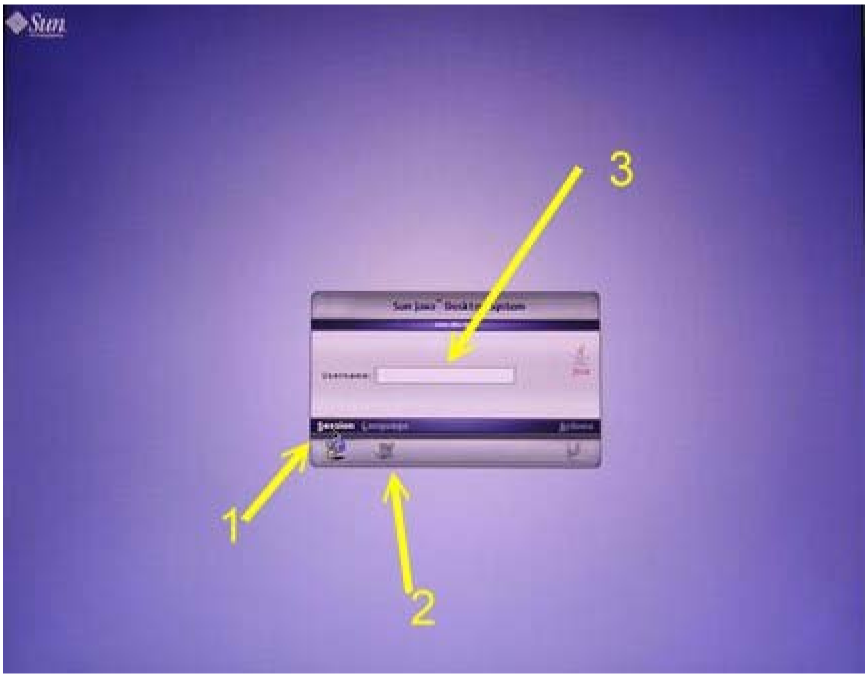


图 78、登入使用 CDE 桌面

在上图当中，箭头 1 的地方你可以选择各个不同的桌面显示方式，当你将鼠标移动到 Session 上头， 然后按下鼠标左键后，就会出现如下的画面：



图 79、登入使用 CDE 桌面

这个画面可以让你选择多种窗口接口啦！鸟哥是建议你直接选择 CDE 的环境，先来感受一下 Solaris 预设的桌面系统，点选 CDE 后，按下『确定』即可回到图 78 的地方；在图 78 的箭头 2 指向 Language (语系) 的地方，点一下鼠标左键，就会出现如下图示：

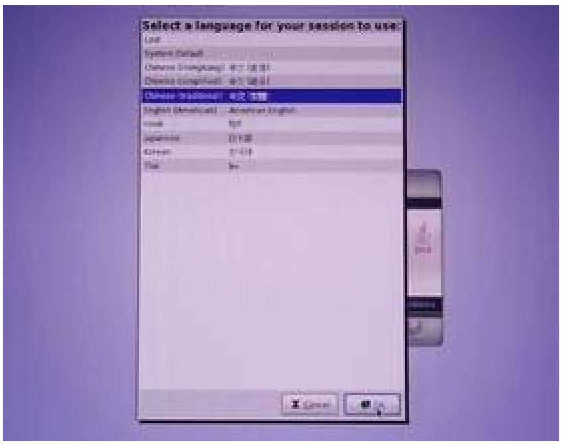


图 80、登入使用 CDE 桌面

上图当中当然是点选繁体中文啊！您说是吧！按下确定后，又回到图 78 的地方，此时填入你的账号， 我们先用 root 这个账号吧！按下 Enter 后，会出现让你输入密码的窗口，输入 root 的密码后， 按下 Enter ，嘿嘿！就会闪过如下图的画面；

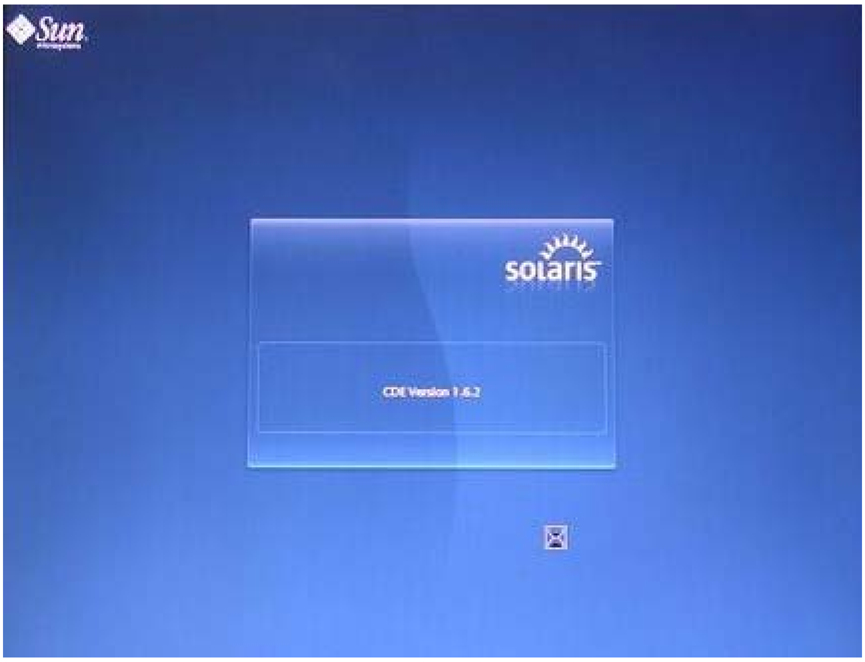


图 81、登入使用 CDE 桌面

然后就出现这个预设的 CDE 桌面的啦！如下图所示，在画面中央下方有出现一堆按钮 (icon)， 在那堆按钮的中央有『一、二、三、四』的字样，那四个按钮是『虚拟桌面』啦， 你可以分别点选看看，会发现每个桌面都有其独特的底部图示呐！然后你可以点选作左边的那个地球， 出现一些网络方面的软件，第一个就是浏览器啰！嘿嘿！其它的功能请自行研究研究的啦！ 桌面很多玩意儿都可以自行处理处理喔！ ^_^



图 82、登入使用 CDE 桌面

另外，在桌面上头任何地方按下鼠标的右键，会出现一堆选单的内容，你也可以每个都给他玩一玩， 而其

中比较重要的是那个『终端机』的接口，你可得将他捉出来，因为那是未来我们最重要的指令下达的接口了！

Solaris 关机

Solaris 的关机是很简单的，不过，你不要直接关掉电源啊！系统会坏掉的！@_@！ 关机的方式是这样的，你可以启动任何一个可以输入指令的终端机，然后在上头直接输入：

```
# init 5
```

下达这个指令之后，大约过 60 秒后，我们的 Solaris 就会开始关闭所有的应用程序，然后自动的关机啦！就这么简单！嘿嘿！以后我们可以好好的来观察一下 Solaris 这个有趣的咚咚了！

参考数据与延伸阅读

- Solaris 简易安装过程：http://www.sun.com/software/solaris/howtoguides/installationhowto.jsp
- Solaris 的多重开机设定：http://www.sun.com/blueprints/0905/819-2889.pdf
- 问题说明：在安装过程当中我们有建立 /home 这个分割区，不过这个分割区在 Solaris 当中是用来处理远程的网络磁盘的， 所以建立这个 partition 反而会造成 Solaris 可能在新增使用者的时候造成无法建立家目录的情况， 那么你挂载在 /home 的那个 partition 可能就无法被使用啊～造成磁盘的浪费～ 此时你可以这样做：

  ```
  # svcadm disable automount
  ```

  或者是自行修改 /etc/automaster 这个档案。不过我们尚未提到很多简易指令的操作， 所以先不谈这个啊！以后有问题时再提出来跟大家说明！

- 问题说明：Solaris 10 与之前旧版的 Solaris 差异性不小喔！包括他的开机流程与服务管理都不太相同了， 所以 Solaris 老手在玩 Solaris 10 时，也要特别留意一些小细节的差异喔！
